THREE-DIMENSIONAL UPWIND SCHEME FOR SOLVING THE EULER EQUATIONS ON UNSTRUCTURED TETRAHEDRAL GRIDS

by

NEAL TILSON FRINK

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Aerospace Engineering

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

September 1991

THREE-DIMENSIONAL UPWIND SCHEME FOR SOLVING THE EULER EQUATIONS ON UNSTRUCTURED TETRAHEDRAL GRIDS

by

Neal Tilson Frink

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Aerospace Engineering

APPROVED:

R. W. Walt

R. W. Walters, Chairman

B. Grossman

J. A. Schetz

Rakesh K. Kapania

R. Kapania

James M. Luckring

J. M. Luckring

September, 1991

Blacksburg, Virginia

THREE-DIMENSIONAL UPWIND SCHEME FOR SOLVING THE EULER EQUATIONS ON UNSTRUCTURED TETRAHEDRAL GRIDS

by

Neal Tilson Frink

Committee Chairman: Robert W. Walters
Aerospace Engineering

(ABSTRACT)

A new upwind scheme is developed for solving the three-dimensional Euler equations on unstructured tetrahedral meshes. The method yields solution accuracy and efficiency comparable to that currently available from similar structured-grid codes. The key to achieving this result is a novel cell reconstruction process which is based on an analytical formulation for computing solution gradients within tetrahedral cells. Prior methodology requires the application of cumbersome numerical procedures to evaluate surface integrals around the cell volume. The result is that higher-order differences can now be constructed more efficiently to attain computational times per cell comparable to those of structured codes.

The underlying philosophy employed in constructing the basic flow solver is to draw on proven structured-grid technology whenever possible in order to reduce risk. Thus, spatial discretization is accomplished by a cell-centered finite-volume formulation using flux-difference splitting. Solutions are advanced in time by a 3-stage Runge-Kutta time-stepping scheme with convergence accelerated to steady state by local time stepping and implicit residual smoothing. The flow solver operates at a speed of 34 microseconds per cell per cycle on a CRAY-2S supercomputer and requires 64 words of memory per cell.

Transonic solutions are presented for a broad class of configurations to demonstrate the accuracy, speed, and robustness of the new scheme. Solutions are shown for the ONERA M6 wing, the Boeing 747-200 configuration, a low-wing transport configuration, a high-speed civil transport configuration, and the space shuttle ascent configuration. Computed surface pressure-coefficient distributions on the ONERA M6 wing are compared with structured-grid results as well as experimental data to quantify the accuracy. A further assessment of grid sensitivity and the effect of convergence acceleration parameters is also included for this configuration. The more complex configurations serve to demonstrate the robustness and efficiency of the new method and its potential for performing routine aerodynamic analysis of full aircraft configurations. For example, the basic transonic flow features are well captured on the space shuttle ascent configuration with only 7 megawords of memory and 142 minutes of CRAY-YMP run time.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his Committee Chairman and Research Advisor, Prof. R. W. Walters for his guidance and support for the duration of this project. Prof. Walters' grasp of fundamental principles and his ability to effectively transfer this knowledge and insight to his students are outstanding.

Many thanks are due Prof. J. A. Schetz, Prof. B. Grossman, Prof. R. Kapania, and Dr. J. M. Luckring for accepting to serve on my Advisory Committee and for providing their useful comments, discussions, and suggestions during the course of this study.

The financial and professional support of the NASA Langley Research Center through its graduate training program is particularly appreciated. Without their support, this work would not have been undertaken.

Special thanks are due to Dr. P. Parikh and Dr. S. Pirzadeh for their willing and essential support in generating the computational grids. I am also grateful to Dr. P. M. Hartwich for generously providing many hours of fruitful discussions, and to Mr. M. D. Salas for his encouragement throughout the endeavor.

Finally, my deepest appreciation is expressed to my wife, Joyce, and children Matthew and Christine for their personal sacrifice. Their continual love and companionship indeed bring the truest satisfaction in life.

TABLE OF CONTENTS

Abstract — ii

Acknowledgements — iv

List of Symbols — vii

1. Introduction — 1

2. Governing Equations — 5

- 2.1 Introduction . . . 5
- 2.2 Mathematical Description . . . 5
  - 2.2.1 Integral Form . . . 5
  - 2.2.2 Differential Form . . . 7
- 2.3 Finite-Volume Discretization . . . 7

3. Upwind Discretization — 11

- 3.1 Introduction . . . 11
- 3.2 Flux Difference Splitting . . . 12
- 3.3 Higher-Order Spatial Differencing . . . 15
  - 3.3.1 Multidimensional Reconstruction . . . 15
  - 3.3.2 Nodal Averaging Procedure . . . 17
- 3.4 Analysis of Reconstruction Scheme . . . 21
  - 3.4.1 Uniform One-Dimensional Stencil . . . 21
  - 3.4.2 Two-Dimensional Cartesian Grid . . . 22
  - 3.4.3 Uniform Triangular Grid . . . 23
- 3.5 Discussion of Limiting . . . 24

4. Time Integration — 26

- 4.1 Introduction . . . 26
- 4.2 Runge-Kutta Time Stepping . . . 26
- 4.3 Convergence Acceleration . . . 28
  - 4.3.1 Local Time Stepping . . . 28
  - 4.3.2 Implicit Residual Smoothing . . . 28

5. Boundary Conditions 30
5.1 Flow Tangency 30
5.2 Far Field 30
6. Programming Strategy 32
6.1 Introduction 32
6.2 Memory Management 32
6.3 Vectorization 34
7. Computational Results 38
7.1 Onera M6 Wing 38
7.1.1 Grid Sensitivity 38
7.1.2 Implicit Residual Smoothing 41
7.2 Boeing 747-200 42
7.3 Low-Wing Transport 43
7.4 High-Speed Civil Transport 44
7.5 Space Transportation System 45
8. Summary and Conclusions 46
Bibliography 48
Figures 56
Appendices 112
Vita 120

LIST OF SYMBOLS

A	area
$\mathbf{A}$	Jacobian matrix
a	speed of sound
$\overline{a}$	local speed of sound determined from Reimann invariants
$A_i^{(x)}, A_i^{(y)}, A_i^{(z)}$	projection of cell i onto x, y, and z planes, respectively
b	span
C_D	drag coefficient
C_L	lift coefficient
C_M	pitching-moment coefficient
C_{RBM}	root-bending moment coefficient
CFL	Courant-Friedrichs-Lewy number
CPU	central processing unit
C_p	pressure coefficient
$\bar{c}$	wing reference chord
$e_\circ$	total energy per unit mass
$\mathbf{F}$	inviscid flux vector of mass, momentum, and energy
$\mathbf{F}_{i,j}$	area-averaged flux of mass, momentum, and energy
FDS	flux difference splitting
FVS	flux vector splitting
$\mid \Delta\tilde{\mathbf{F}} \mid_{1,4,5}$	components of inviscid flux associated with distinct eigenvalues
$\mathbf{f}, \mathbf{g}, \mathbf{h}$	inviscid flux vectors in Cartesian coordinates
f	test function
f_x, f_y, f_z	derivative of f with respect to $x, y,$ and$z$
$HSCT$	high-speed civil transport
$h_\circ$	total enthalpy per unit mass
IRS	implicit residual smoothing
$\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$	x, y, and z components of a unit vector

L	length
$\mathbf{L}$	Cartesian length vector
L_2 norm	RMS average of mass, momentum and energy residuals
LWT	low-wing transport
l	edge length of triangle
M_∞	Mach number
N	total number of nodes or vertices
$\hat{\mathbf{n}}$	outward facing unit normal vector
n_x, n_y, n_z	direction cosines of unit normal vector
p	pressure
$\Delta p/p_\infty$	normalized pressure difference
$\mathbf{Q}$	vector of conserved variables
$\mathbf{Q}_i$	volume-averaged state of conserved variables
$\Delta\mathbf{Q}$	$\equiv \mathbf{Q}_R - \mathbf{Q}_L$
$\mathbf{q}$	vector of primative variables
q	scalar quantity
RMS	root-mean-square
$\mathbf{R}$	residual
$\overline{\mathbf{R}_i}$	vector of Laplacian-averaged residual terms
$R^\pm$	Riemann invariants
$\Delta\mathbf{r}$	radius vector
Δr	magnitude of radius vector
S	entropy
STS	Space Transportation System
$\tilde{\mathbf{T}}, \tilde{\mathbf{T}}^{-1}$	right and left eigenvectors, respectively
t	time
U	contravariant velocity
ΔU	contravariant velocity difference
u, v, w	Cartesian velocities

$\mathbf{V}$	velocity vector
V	volume
w_i	weighting factor
x, y, z	Cartesian coordinates
α	angle of attack
γ	ratio of specific heats
ε	small parameter
η	nondimensional span location on wing
κ	cell face
$\mathbf{\Lambda}$	diagonal matrix of eigenvalues
ρ	density
∇	gradient operator
∇^2	Laplacian operator

Subscripts:

b	domain boundary
c	cell volume
f	cell face
i	i^{th} cell volume
int	interior
j	j^{th} cell face index
L	left
n	node of a tetrahedral cell
o	stagnation
R	right
ref	reference
κ	cell face
∞	freestream condition

Superscripts:

$\tilde{}$	Roe-averaged quantity

1. INTRODUCTION

The role of Computational Fluid Dynamics (CFD) in maintaining the competitiveness of the U.S. aircraft industry in the international marketplace was a recurring theme at the 1991 NASA CFD Conference [1]. Panel discussions held between industry, university, and government researchers and managers drew a clear consensus that advanced CFD codes were now considered principle technology in the analysis and design of air vehicles. From the discussions came a further consensus that more advanced CFD tools were needed by industry which would enable the working-level engineer to provide final answers to a problem within days. The industry personnel expressed a desire that more effort be directed by government and university research organizations toward bridging the gap between research on basic algorithms and the development of application software.

The primary need reiterated during the discussions was for user-friendly software capable of providing timely analysis and design of complex aircraft configurations across the speed range from subsonic to supersonic. Computational tools have been maturing rapidly toward this goal over the past decade. The earliest computations on geometrically complex aircraft geometries were performed with either one of two approaches. The first, and simplest, utilized an extended small disturbance-type equation coupled with mean-surface boundary conditions [2-4]. The second, and more sophisticated approach employed the full potential equation coupled with surface-conforming boundary conditions [5-7]. Due to the simplicity of the boundary conditions, the small disturbance method could be more readily applied to aircraft configurations with nacelles, pylons, winglets, and canards [8]. However, the assumptions of isentropic and irrotational flow employed in both methods were not strictly correct when shock waves are present, and thus, limited their accuracy.

As more powerful supercomputers became available, interest increased in developing efficient and accurate algorithms for solving the Euler and Navier-Stokes equations. The Euler equations provide a more refined model of the inviscid flow by allowing entropy rises through shock waves while conserving mass, momentum, and energy, and by admitting vorticity. The Navier-Stokes equations include the additional terms for the viscous shear stresses. Early applications of the Euler equations to complex geometries were made using single-block grids [9, 10]. These efforts were continued and extended to viscous flow over complex geometries [11].

As Euler and Navier-Stokes algorithms matured, efforts were also underway to extract more geometrical flexibility from the computational grid. Domain decomposition became a preferred approach for solving the flexibility problem. This approach involves dividing the spatial domain around a complex geometry into zones of simpler grid blocks, then solving the flow equations independently within each block while maintaining communication across the block boundaries. The key obstacle to overcome was to provide an efficient and accurate transfer of information across zonal boundaries. Efficient schemes have been developed for patching dissimilar interfaces along a common boundary in a conservative manner [12 to 16], and for overlapping the boundaries of independent component grids [17,18]. Grid embedding techniques were also under investigation as a means of increasing local resolution of flow features. [19 to 21]. Domain decomposition methodologies have been successfully applied for the computation of the viscous flow around complex configurations such as the F-16 [22], STS ascent configuration [23], and the F/A-18 [24], F/A-18 forebody [25,26], and a high-speed accelerator configuration [27]. However, even with the recent dramatic progress in applying the various domain decomposition techniques to complex configurations, the grid setup time is still measured in months. Thus, more advances are necessary before such large-scale computations are performed routinely.

New opportunities for the analysis and design of complex configurations are emerging through the relatively new technology of unstructured grids [28-44]. Tetrahedral cells offer exceptional geometric flexibility in constructing quality grids around complex configurations without resorting to the large scale decomposition techniques. Furthermore, the time required to generate an unstructured tetrahedral grid is significantly less than that for a structured zonal grid. Unstructured flow solvers utilize a random data structure which better facilitates the adaptation of the grid to the physics of the flow, or the movement of components within the grid, since grid points can be placed where ever needed.

While unstructured methodology is rooted in the Finite Element Method (FEM) from the discipline of Structural Analysis [45], a variety of algorithms have been developed for solving the compressible flow equations. Adaptations of the FEM have been successfully applied to the flow equations by some researchers [30 to 37]. Others have employed more conventional central-differenced finite-volume techniques with success [28,29]. More recently, upwind-differenced finite-volume methods have been investigated for unstructured grids [38 to 44]. While many of these methods have shown varying degrees of success, most fall well short of their structured counterparts in terms of efficiency and accuracy. Thus, significant advancements are needed before this methodology will be used widely for solving practical three-dimensional problems.

There is a need for more fundamental research in both unstructured solution algorithms and grid generation methodology. Current solution algorithms are much less efficient than structured ones due to the indirect addressing required by random data sets. Furthermore, many of the unstructured algorithms have not demonstrated sufficient accuracy for addressing realistic problems, in particular for viscous flow. The efficient implementation of viscous terms and turbulence models on arbitrary tetrahedral cells has yet to be resolved satisfactorily, although efforts are underway to address this problem [34,36]. The generation of tetrahedral grids is a research topic in itself. The most common approaches are the Advancing Front technique [46] and Delaunay Triangulation [47,48]. Methods for generating highly-stretched viscous grids on complex three-dimensional geometries are in the very early stages of development [49,50], but are pivotal to the overall advancement of unstructured grid methodology. Solution adaptive gridding techniques are presently under investigation. While some have been applied in three-dimensions [30, 33, 35, and 37], additional work is needed before realizing the full benefits on complex configurations.

More applied research is needed in order to focus the development of unstructured methodology toward a useable tool in a timely fashion. Application of the basic methodology to solving relevant problems tends to expose deficiencies at an early stage and accelerate the implementation of other useful features such as design algorithms or real-gas models. Documented code calibration studies are essential for evaluating solution accuracy and establishing confidence within the user community. These studies should be conducted carefully for a broad class of geometries for which experimental data is available.

The long-term goal for developing unstructured grid methodology is to produce a useful engineering tool which ultimately can provide rapid analysis of full aircraft configurations at flight Reynolds numbers. This goal is sufficiently broad to require the concerted effort of many researchers in bringing it to fruition. The objective of the present work is to construct an accurate and efficient unstructured three-dimensional inviscid flow solver. The coding is constructed as a modular platform on which to build new capabilities. The underlying development philosophy is to draw on proven structured-grid technology whenever possible in order to reduce risk. The resulting scheme is a cell-centered finite volume formulation which is applied to tetrahedral cells using flux-difference splitting and explicit time integration with convergence acceleration. An accurate and efficient higher-order differencing scheme for tetrahedral cells was not available in the literature and could not be readily extended from structured methodology, thus, requiring the development of substantially new technology. This development constitutes one of the major contributions of the present work, and has been summarized in Refs. [43, 44]. In order to minimize resource requirements, particular attention is given to both memory management and vectorization. All grids are generated using an extended version of the advancing front grid generator, VGRID3D [51]. Graphic postprocessing of solutions was performed using VPLOT3D described in Ref. [51].

2. GOVERNING EQUATIONS

2.1 Introduction

The fluid motion is to be governed by the time dependent Euler equations for an ideal gas which are a coupled set of equations that express the conservation of mass, momentum, and energy. The general assumptions employed are that the fluid is compressible, inviscid, nonconducting, adiabatic, and is not influenced by body forces. Since the equations of fluid motion are based upon conservation laws, it is convenient to express them in a form which clearly displays the conserved quantities. The governing equations will be expressed in conservative form using both the integral and differential formulations.

2.2 Mathematical Description

2.2.1 Integral Form

The integral form of the governing equations can be derived from first principles using a control-volume approach that is based on the satisfaction of macroscopic physical laws. This approach is particularly useful when dealing with inviscid continuum flow in which discontinuities exist, e.g. shock waves or contact discontinuities. The integral form of the flow equations are equivalent to the differential ones in regions of smooth flow and are also valid across discontinuities where they reduce to the Rankine-Hugoniot jump relations [52]. The integral form is especially useful when dealing with finite-volume discretization techniques, as will be discussed in a subsequent section.

The governing equations presented below express a relationship where the time rate of change of the state vector $\mathbf{Q}$ within the domain Ω is balanced by the net flux $\mathbf{F}$ across the boundary surface $\partial\Omega$

$$\int\int\int_{\Omega} \frac{\partial \mathbf{Q}}{\partial t} dV + \int\int_{\partial\Omega} \mathbf{F}(\mathbf{Q}) \cdot \hat{\mathbf{n}} dA = \mathbf{0} \tag{2.1}$$

where the conserved variables are

$$\mathbf{Q}=\left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho e_\circ\end{array}\right\}=\left\{\begin{array}{c}\text{density}\\ x-\text{momentum per unit volume}\\ y-\text{momentum per unit volume}\\ z-\text{momentum per unit volume}\\ \text{total energy per unit volume}\end{array}\right\}$$

and the flux vector is

$$\mathbf{F}(\mathbf{Q})\cdot\hat{\mathbf{n}}=(\mathbf{V}\cdot\hat{\mathbf{n}})\left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho h_\circ\end{array}\right\}+p\left\{\begin{array}{c}0\\ \hat{n}_x\\ \hat{n}_y\\ \hat{n}_z\\ 0\end{array}\right\}.$$

The terms in Eq. (2.1) are nondimensionalized as follows: noting that the superscript * denotes a dimensional quantity and the subscript ∞ represents freestream conditions, then $\rho = \rho^*/\rho^*_\infty$, $u = u^*/a^*_\infty$, $v = v^*/a^*_\infty$, $w = w^*/a^*_\infty$, and $e_\circ = e^*_\circ/(a^*_\infty)^2$. The parameters $\hat{n}_x, \hat{n}_y$, and $\hat{n}_z$ are the Cartesian components of the *exterior* surface unit normal $\hat{\mathbf{n}}$ on the boundary $\partial\Omega$. The Cartesian velocity components are u, v, and w in the x, y, and z directions, respectively. It is necessary to define an equation of state in order to close the system of equations. Using the ideal gas assumption, the pressure can be expressed as

$$p=\rho(\gamma-1)[e_\circ-\frac{1}{2}(u^2+v^2+w^2)] \tag{2.2}$$

where γ is the ratio of specific heats and is prescribed as 1.4 for air. The total enthalpy is defined as

$$h_\circ=\frac{\gamma}{\gamma-1}\frac{p}{\rho}+\frac{1}{2}(u^2+v^2+w^2). \tag{2.3}$$

2.2.2 Differential Form

It is often convenient to use the differential form of the governing equations for discussion purposes. This form of the equations relies on aspects of continuum mathematics in the derivation and is restricted by the assumptions that the domain must have continuous partial derivatives and be bounded by a simple, piecewise smooth surface. The differential form of the equations can be derived by applying Gauss's theorem to the flux integral in Eq. (2.1). It is in the application of Gauss's theorem that the restrictive continuum properties are introduced.

The resulting flow equations in a conservative, Cartesian, differential form are:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z} = \mathbf{0} \tag{2.4}$$

with

$$\mathbf{Q} = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_\circ \end{array} \right\}$$

and

$$\mathbf{f} = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho u h_\circ \end{array} \right\}, \mathbf{g} = \left\{ \begin{array}{c} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho v h_\circ \end{array} \right\}, \mathbf{h} = \left\{ \begin{array}{c} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho w h_\circ \end{array} \right\}$$

As before, these equations are closed using the ideal gas assumption described in Eq. (2.2). While Eqs. (2.4) are shown in mathematical *conservation form*, the continuum assumption employed in their derivation precludes the *conservative property* across discontinuities within the domain.

2.3 Finite-Volume Discretization

The finite-volume technique is a discretized application of the control-volume approach that was used in the derivation of the integral form of the governing equations. The global domain is divided into a finite number of subdomains, each one of arbitrary volume Ω_i and closed by a boundary $\partial\Omega_i$, where Eq. (2.1) is applied to each subdomain. Since each subdomain Ω_i shares common boundaries with its neighbors, this approach retains the *conservative property* inherent to the integral equations. This feature occurs because the flux which exits across an interior boundary of one subdomain will enter the neighboring subdomain across the common boundary. The end result is that the contributions from the fluxes across all of the *interior* boundaries within the global domain will exactly cancel each other, leaving only those flux contributions across the *external* boundary.

Numerical approximations to the the volume and surface integrals of Eq. (2.1) lead to the unknown states being interpreted as *volume-averaged* values in each subdomain. The volume-averaged values for the conserved variables $\mathbf{Q}$ are

$$\langle \mathbf{Q}_i \rangle = \frac{1}{V_i} \int\int\int_{\Omega_i} \mathbf{Q} dV. \tag{2.5}$$

By making the assumption of a fixed subdomain so that the time derivative may be brought outside the integral in Eq. (2.1), and then substituting Eq. (2.5), a semi-discrete approximation to the equations can be written

$$V_i \frac{\partial \mathbf{Q}_i}{\partial t} + \sum_{j=1}^{\kappa(i)} \mathbf{F}_{i,j} \Delta A_{i,j} = \mathbf{0}. \tag{2.6}$$

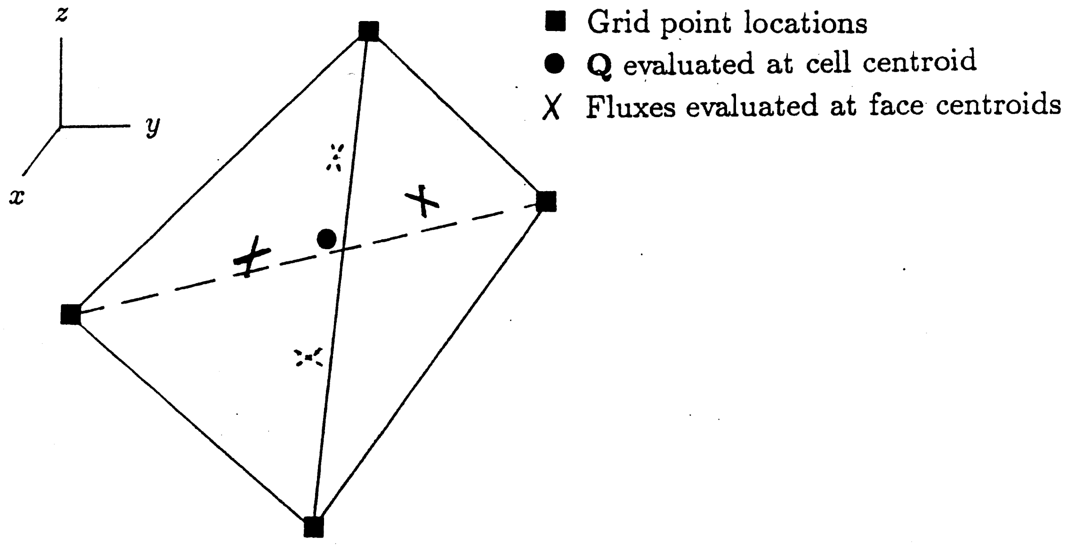
where

$$\mathbf{Q}_i \equiv \langle \mathbf{Q}_i \rangle$$

Equation (2.6) states that the time rate of change of the *volume-averaged* state $\mathbf{Q}$ in the ith control volume is balanced by the summation of the *area-averaged* flux $\mathbf{F}_{i,j}$ through the discrete boundary faces κ. Equation (2.6) can also be interpreted as a discrete form of the differential equation if $\mathbf{Q}_i$ is assumed to be the state at the nodal point i.

The subdomains Ω_i can be represented by any arbitrarily shaped volume, i.e. hexahedrals, tetrahedrals, polygons, etc. For *structured* methods, that volume is usually prescribed as a hexahedral cell where the Cartesian coordinates can be transformed into a generalized three-component curvilinear frame of reference. The summation in Eq. (2.6) is then applied over the six faces of the cell. Many efficient algorithms which are based on a logical indexing of data structure are available [53-65] for solving the resulting system of algebraic equations. The system of equations can also be solved using generalized indexing schemes which are often referred to in literature as *unstructured* techniques [66].

The present method employs a generalized indexing scheme with a *cell-centered* control volume. The global domain is divided into tetrahedral *cells* where each cell is viewed as a control volume Ω_i that is enclosed by the four triangular surfaces $\partial\Omega_i$, as shown in sketch (a)


z
y
x
■ Grid point locations
● **Q** evaluated at cell centroid
X Fluxes evaluated at face centroids

Sketch (a)

Equation (2.6) is applied over the four faces of each cell. The equations for the metric terms of tetrahedral cells are presented in Appendix A1.

The cell-centered approach contrasts with the more commonly used *node-centered* approach [28, 30, 31, 33, 36, 37, 40] that utilizes polygonal control volumes which are constructed around the vertices of the tetrahedra. The primary reason for choosing the cell-centered approach is due to the increased spatial resolution afforded by the scheme on a set grid. The higher spatial resolution stems from the feature that tetrahedral grids contain between five and six times more *cells* than *nodes*. Similarly, the surface resolution is doubled since there are twice as many boundary faces as boundary nodes. Secondary reasons for choosing the cell-centered approach are that it is more straightforward to program and to treat the boundary conditions accurately.

3. UPWIND DISCRETIZATION

3.1 Introduction

The finite volume approach has the basic feature that the time integration and spatial discretization procedures are fully decoupled. Thus, the spatial discretization is accomplished independent of the time integration. The problem to be addressed in this section pertains to the estimation of fluxes across the faces of the tetrahedral control volumes. Once these fluxes have been determined as a function of the state variable $\mathbf{Q}$, the solution can be advanced in time by any number of time integration algorithms.

The interface fluxes are generally constructed in one of two manners: central differencing or upwind differencing. For central differencing, the numerical flux $\mathbf{F}(\mathbf{Q}_L, \mathbf{Q}_R)$ is computed by averaging the fluxes corresponding to $\mathbf{Q}_L$ and $\mathbf{Q}_R$. This approach leads to a decoupling of adjacent cells and is inherently unstable. To achieve stability, artificial dissipation must be added. On structured grids it has been found that an effective and inexpensive dissipation formula can be constructed as a blend of second and fourth differences in the flow variables [67]. This has been extended to unstructured tetrahedral grids by constructing the dissipative operator as a blend of a Laplacian and a biharmonic operator [68], corresponding to the second and fourth differences respectively, and using the finite volume or integral approximation to evaluate these operators.

Upwind schemes determine the interface fluxes based on the characteristic theory for hyperbolic systems of equations. Operators are constructed so that the differencing is performed *upwind* or in the direction opposite to that in which the components of information are traveling. This approach has the advantage of being more robust and requiring less user interaction than does the central difference technique. There are generally two classes of upwind methods: flux-vector splitting (FVS) and flux-difference splitting (FDS). The more popular FVS schemes are those of Steger and Warming [69] and Van Leer [70]. The most popular FDS technique is Roe's scheme [71].

The basic idea behind FVS techniques is to split the inviscid flux vector in one space dimension into two parts, according to the sign of the eigenvalues, each of which contain the information that propagates downstream and upstream, respectively. The two parts are then differenced separately in a stable manner using a one-sided extrapolation formula consistent with the direction of propagation. In general, downstream propagating information is extrapolated from the upstream direction, and conversely for the upstream propagating information.

The FDS technique does not split the flux vector but reconstructs the fluxes by determining an approximate solution to a Reimann problem [71]. For that, discontinuous states are assumed to exist on either side of a cell interface. An approximate solution for this condition is written in terms of waves propagating upstream and downstream relative to the cell interface, each of which is associated with a distinct eigenvalue. For the Euler equations, the solution consists of three waves centered at the cell interface, i.e. a shock wave, a contact discontinuity, and an expansion wave. In general, FDS is considered more accurate and provides a sharper resolution of shocks and contact discontinuities than does FVS. A good review of the various flux formulas is presented in Ref. [72]. The FDS approach will be employed in the present method.

3.2 Flux Difference Splitting

The philosophy behind Roe's approach to FDS is to compute approximate solutions to the underlying set of Riemann problems which still describe the important nonlinear behavior of the interacting waves. This contrasts with Godunov's approach [73] which is to obtain a more expensive exact nonlinear solution to what is already an approximation of the data. Roe's scheme exploits the fact that the Riemann solution for any set of *linear* conservation laws is easily computed. Its derivation is based only on a one-dimensional interaction of characteristic waves, but can be applied in multidimensions if the assumption is made that conserved quantities in grid cells are exchanged by waves traveling normal to the cell interfaces, i.e. locally one-dimensional. Hence, velocities parallel to the cell interface are ignored and differences in the parallel components are assumed to occur across the contact surface. Research is also underway on a truly multidimensional FDS approach [74 to 76], but this has not been applied in the present work. A good derivation of Roe's one-dimensional scheme is presented in Ref. [53] along with the extension to three dimensions. Key elements of the derivation will be presented in the following.

The flux across each cell face κ is computed using the numerical flux formula

$$\mathbf{F}_\kappa = {}^1\!/_2[\mathbf{F}(\mathbf{Q}_L) + \mathbf{F}(\mathbf{Q}_R) - \mid \tilde{\mathbf{A}} \mid (\mathbf{Q}_R - \mathbf{Q}_L)]_\kappa \tag{3.1}$$

Here $\mathbf{Q}_L$ and $\mathbf{Q}_R$ are the conserved variables to the left and right of the interface κ. Eq. (3.1) essentially expresses the central difference of the fluxes plus an upwind correction. For a central-difference algorithm, the upwind correction term would be replaced by the artificial dissipation terms.

Since the Euler equations are nonlinear, and Roe's scheme is based on linear concepts, the equations are *linearized* by evaluating the Jacobian matrix $\mathbf{A} \equiv \partial\mathbf{F}/\partial\mathbf{Q}$ with the averaged quantities:

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \\
\tilde{u} &= (u_L + u_R\sqrt{\rho_R/\rho_L})/(1 + \sqrt{\rho_R/\rho_L}) \\
\tilde{v} &= (v_L + v_R\sqrt{\rho_R/\rho_L})/(1 + \sqrt{\rho_R/\rho_L}) \\
\tilde{w} &= (w_L + w_R\sqrt{\rho_R/\rho_L})/(1 + \sqrt{\rho_R/\rho_L}) \\
\tilde{h}_\circ &= (h_{\circ_L} + h_{\circ_R}\sqrt{\rho_R/\rho_L})/(1 + \sqrt{\rho_R/\rho_L}) \\
\tilde{a}^2 &= (\gamma - 1)(\tilde{h}_\circ - (\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2)/2)
\end{aligned} \tag{3.2}$$

Here, the Roe-averaged matrix $\tilde{\mathbf{A}}$ (note tilde) is a mean value of $\mathbf{A}$ with the properties

(i) $\tilde{\mathbf{A}}(\mathbf{Q}_L, \mathbf{Q}_R) \to \mathbf{A}(\mathbf{Q})$ as $\mathbf{Q}_L \to \mathbf{Q}_R \to \mathbf{Q}$;

(ii) $\tilde{\mathbf{A}}(\mathbf{Q}_L, \mathbf{Q}_R)[\mathbf{Q}_R - \mathbf{Q}_L] = \mathbf{F}(\mathbf{Q}_R) - \mathbf{F}(\mathbf{Q}_L)$;

(iii) $\tilde{\mathbf{A}}$ has a complete set of real eigenvalues and eigenvectors.

Property (i) ensures consistency of the governing differential equation so that the approximate solution tends to the exact solution for small differences in data across the interface. Property (ii) ensures that $\tilde{\mathbf{A}}$ satisfies the Rankine-Hugoniot shock jump condition and is responsible for the sharp resolution of steady shock waves. Property (iii) ensures that the matrix $\tilde{\mathbf{A}}$ has three independent eigenvalues which allows the matrix to be written in the canonical form:

$$| \tilde{\mathbf{A}} | = \tilde{\mathbf{T}} \, | \tilde{\boldsymbol{\Lambda}} | \, \tilde{\mathbf{T}}^{-1} \tag{3.3}$$

where $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{T}}^{-1}$ are the right and left eigenvectors, respectively, and $\tilde{\boldsymbol{\Lambda}}$ is the diagonal matrix of eigenvalues.

The Roe-averaged variables in Eq. (3.2) were derived in Ref. [77] to satisfy the three properties simultaneously. Properties (i) and (iii) could be satisfied by virtually any simple algebraic average of the left and right states, but property (ii) can not. The special *square-root averaging* evident in Eq. (3.2) is necessary to satisfy the second property, i.e. the Rankine-Hugoniot shock jump condition..

Using Eq. (3.3) to rewrite the last term in Eq. (3.1) as:

$$| \tilde{\mathbf{A}} | (\mathbf{Q}_R - \mathbf{Q}_L) = \tilde{\mathbf{T}} \, | \tilde{\boldsymbol{\Lambda}} | \, \tilde{\mathbf{T}}^{-1} \Delta \mathbf{Q} \tag{3.4}$$

it can be reduced to three $\Delta\mathbf{F}$ flux components, each of which is associated with a distinct eigenvalue:

$$\tilde{\mathbf{T}} \, | \tilde{\boldsymbol{\Lambda}} | \, \tilde{\mathbf{T}}^{-1} \Delta \mathbf{Q} = | \Delta \tilde{\mathbf{F}}_1 | + | \Delta \tilde{\mathbf{F}}_4 | + | \Delta \tilde{\mathbf{F}}_5 | \tag{3.5}$$

with

$$| \Delta \tilde{\mathbf{F}}_1 | = | \tilde{U} | \left\{ \left(\Delta\rho - \frac{\Delta p}{\tilde{a}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \frac{\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2}{2} \end{bmatrix} + \tilde{\rho} \begin{bmatrix} 0 \\ \Delta u - \hat{n}_x \Delta U \\ \Delta v - \hat{n}_y \Delta U \\ \Delta w - \hat{n}_z \Delta U \\ \tilde{u}\Delta u + \tilde{v}\Delta v + \tilde{w}\Delta w - \tilde{U}\Delta U \end{bmatrix} \right\}$$

$$| \Delta\tilde{\mathbf{F}}_{4,5} | = | \tilde{U} \pm \tilde{a} | \left(\frac{\Delta p \pm \tilde{\rho}\tilde{a}\Delta U}{2\tilde{a}^2}\right) \begin{bmatrix} 1 \\ \tilde{u} \pm \hat{n}_x \tilde{a} \\ \tilde{v} \pm \hat{n}_y \tilde{a} \\ \tilde{w} \pm \hat{n}_z \tilde{a} \\ \tilde{h}_\circ \pm \tilde{U}\tilde{a} \end{bmatrix}$$

where $\tilde{U} = \tilde{u}\hat{n}_x + \tilde{v}\hat{n}_y + \tilde{w}\hat{n}_z$ and $\Delta U = \hat{n}_x \Delta u + \hat{n}_y \Delta v + \hat{n}_z \Delta w$.

For a first-order scheme, the state of the primitive variables at each cell face is set to the cell-centered averages on either side of the face.

3.3 Higher-Order Spatial Differencing

3.3.1 Multidimensional Reconstruction

The challenge posed in constructing an effective higher-order scheme is to determine an accurate estimate of the left and right states at the cell faces for the flux calculation. One approach [40, 41] would be to extend the MUSCL-differencing technique, which is widely used in structured algorithms, to unstructured grids. Since this procedure involves colinear interpolations across arbitrary tetrahedral cells, accurate results can be difficult to attain. An alternate approach proposed by Barth and Jespersen [38] is to perform a multidimensional linear reconstruction of cell-averaged data. In the reconstruction approach, higher-order accuracy is achieved by expanding the cell-centered solution to each cell face with a Taylor series:

$$\mathbf{q}(x, y, z) = \mathbf{q}(x_c, y_c, z_c) + \nabla \mathbf{q}_c \cdot \Delta \mathbf{r} \; + \; O(\Delta r^2) \tag{3.6}$$

where

$$\mathbf{q} \equiv [\rho, u, v, w, p]^T$$

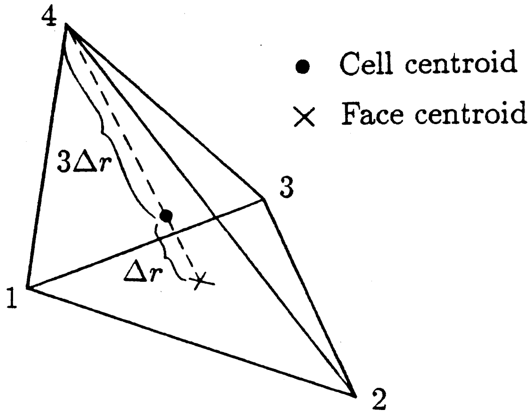
This formulation requires that the solution gradient $\nabla \mathbf{q}$ be determined at the cell centroids. The evaluation of this term is generally quite expensive and can consume a large fraction of the total computational time. Barth and Jespersen proposed three approaches [38] in two-dimensions for computing the gradient. However, the most accurate of the approaches is not easily extendible to three dimensions, and the least accurate ones utilize only a portion of the surrounding information. What is desired is an inexpensive approach which produces a truly multidimensional reconstruction, yet is relatively simple to apply in three dimensions.

A new scheme containing these properties was proposed in Ref. [43] for estimating the solution gradient. The general approach was to: 1) coalesce surrounding cell information to the vertices or nodes of the candidate cell, then 2) apply the mid-point trapezoidal rule to evaluate the surface integral of the gradient theorem

$$\nabla \mathbf{q_c} = \frac{1}{V_{\Omega_i}} \oint_{\partial\Omega_i} \mathbf{q}\hat{\mathbf{n}} dA \tag{3.7}$$

over the four faces of each tetrahedral cell. Here, V_{Ω_i} denotes the volume enclosed by the surface Ω_i. Results presented in Ref. [43] demonstrated that this scheme could achieve accuracies comparable to those obtained with structured upwind codes, with approximately twice the CPU time/cell.

Additional analysis of that scheme has resulted in a simplification which further reduces the run time by one-half by eliminating the need to explicitly evaluate the integral in Eq. (3.7). This new scheme is analytically equivalent to that in Ref. [43], and is formally derived in Appendix A2 for two dimensions. The simplification stems from some useful geometrical invariant features of triangles and tetrahedra. These features are illustrated for an arbitrary tetrahedral cell in sketch (b).



Sketch (b)

Note that a line extending from a cell-vertex through the cell-centroid will always intersect the centroid of the opposing face, which is precisely where the area-averaged flux is evaluated in Eq. (3.1). Furthermore, the distance from the cell-vertex to the cell-centroid is always three-fourths of that from the vertex to the opposing face. By using these invariants along with the fact that Δr is the distance between the cell centroid and the face centroid, the second term in Eq.(3.6) can be evaluated as:

$$\begin{aligned}\nabla \mathbf{q_c} \cdot \mathbf{\Delta r} &= \frac{\partial \mathbf{q}}{\partial r}\Delta r \\ &\approx \left[\frac{{}^1\!/\!_3(\mathbf{q}_{n_1} + \mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - \mathbf{q}_{n_4}}{4\Delta r}\right]\Delta r\end{aligned} \tag{3.8}$$

Substituting Eq. (3.8) into Eq. (3.6) results in a simple, universal expression for a Taylor series expansion within a tetrahedral cell:

$$\mathbf{q}_{f_{1,2,3}} = \mathbf{q}_c + {}^1\!/\!_4\,[{}^1\!/\!_3(\mathbf{q}_{n_1} + \mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - \mathbf{q}_{n_4}] \tag{3.9}$$

where as illustrated in sketch (b), the subscripts n_1, n_2, n_3 denote the nodes comprising face $f_{1,2,3}$ of cell c and n_4 corresponds to the opposite node. As shown in appendix A2, Eq. (3.9) is formally second-order accurate. The only question of accuracy in the overall scheme lies in the nodal averaging procedure. Some analysis of this procedure is given in the next section, but no formal estimate of accuracy has yet been determined.

3.3.2 Nodal-Averaging Procedure

As observed in Eq. (3.9), the state at the cell faces, $\mathbf{q}_f$, are a function of the cell-averaged state, $\mathbf{q}_c$, and that of the four nodes, $\mathbf{q}_n$. The cell-averaged states are assumed known from the solution. Estimates of the state at the nodes can be determined from the surrounding cell-averaged values using a weighted averaging process which incorporates available geometric information. A multidimensional weighted averaging of surrounding cell-centered data to a node can be achieved by an expression of the form

$$\mathbf{q}_n = \Big(\sum_{i=1}^{N} \mathrm{w}_i \mathbf{q}_{c,i}\Big) / \Big(\sum_{i=1}^{N} \mathrm{w}_i\Big) \tag{3.10}$$

The subscripts n and c, i refer to the node and surrounding cell-centered values, respectively. Numerical experiments were performed using known test functions to determine a satisfactory definition of the weighting factor w_i. An arbitrary tetrahedral test grid was constructed within a unit cube as shown in Fig. 1. Both linear and nonlinear test functions were used to define a known state and its gradient at the cell centroids. The linear function used is

$$f(x,y,z)_{linear} = x + y + z \tag{3.11a}$$

where the gradients are

$$f_x = f_y = f_z = 1.0 \tag{3.11b}$$

The nonlinear function used is

$$f(x,y,z)_{nonlinear} = x^2 + \sin(\pi y) + e^{2z} \tag{3.12a}$$

where the gradients are

$$f_x = 2x, \quad f_y = \pi\cos(\pi y), \quad f_z = 2e^{2z} \tag{3.12b}$$

Several weighting factors of the form

$$\mathrm{w}_i = V_i,\ r_i,\ \frac{1}{r_i},\ \text{and}\ \frac{1}{r_i^2} \tag{3.13}$$

where V_i is cell volume and

$$r_i = [(x_{c,i} - x_n)^2 + (y_{c,i} - y_n)^2 + (z_{c,i} - z_n)^2]^{\frac{1}{2}} \tag{3.14}$$

were explored to determine which would produce the least total error in computing the gradient. The error was computed as a summation of the individual cell errors

$$\text{Error} = \frac{1}{3N_{cells}} \sum \mid f'_{exact} - f'_{computed} \mid_{x,y,z} \tag{3.15}$$

The results of that exercise are presented in Table 1.

Table 1. Errors in computing gradient for test-functions.

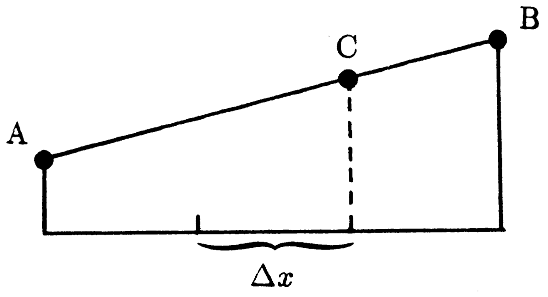
w_i	Linear Function	Nonlinear Function
V_i	0.1059	0.4820
r_i	0.0419	0.3018
$1/r_i$	0.0344	0.2621
$1/r_i^2$	0.0480	0.2842

As can be observed, the form $\mathrm{w}_i = 1/r_i$ produces the lowest total error for each of the three components. The resulting nodal averaging formula is

$$\mathbf{q}_n = \Big(\sum_{i=1}^{N} \frac{\mathbf{q}_{c,i}}{r_i}\Big) \Big/ \Big(\sum_{i=1}^{N} \frac{1}{r_i}\Big). \tag{3.16}$$

In constructing this formula, it is assumed that the known values of the solution are concentrated at the cell centroids, and that the contribution to a node from the surrounding cells is inversely proportional to the distance from each centroid to the node. Note that this new reconstruction process incorporates information from all of the cells surrounding the candidate cell, thus producing a truly multidimensional higher-order expansion in Eq. (3.9). Furthermore, the method is extremely easy to apply in either two-dimensions or three-dimensions. The primary information which must be supplied are which cells surround each node. For boundary nodes, the surrounding face-centered boundary conditions and respective distances are used in Eq. (3.16).

Equation (3.16) is essentially a multidimensional linear interpolation formula. For example, consider in sketch (c) the one-dimensional interpolation between states A and B to point C:

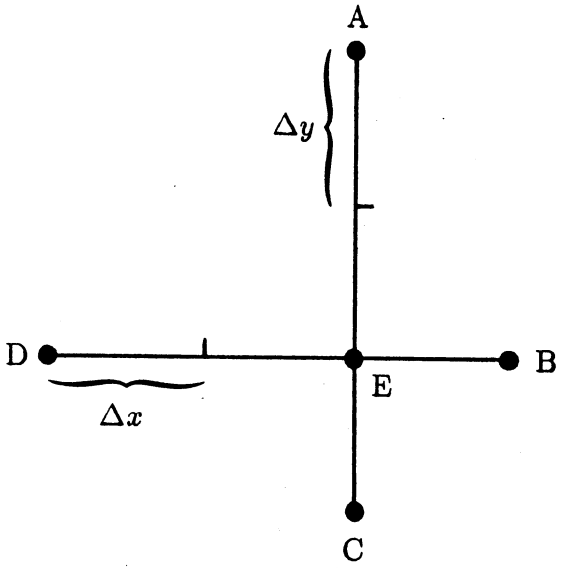


Sketch (c)

Apply Eq. (3.16) as

$$
\begin{aligned}
q_C &= \left(\frac{q_A}{2\Delta x} + \frac{q_B}{\Delta x}\right) \Big/ \left(\frac{1}{2\Delta x} + \frac{1}{\Delta x}\right) \\
&= {}^1\!/_3 q_A + {}^2\!/_3 q_B
\end{aligned}
\tag{3.17}
$$

In the next example, Eq. (3.16) can be characterized as an average of two linear interpolations. Consider the application of Eq. (3.16) to the following:



Sketch (d)

$$
\begin{aligned}
q_E &= \Big(\frac{q_A}{2\Delta y} + \frac{q_C}{\Delta y} + \frac{q_B}{\Delta x} + \frac{q_D}{2\Delta x}\Big) \Big/ \Big(\frac{1}{2\Delta y} + \frac{1}{\Delta y} + \frac{1}{\Delta x} + \frac{1}{2\Delta x}\Big) \\
&= \frac{2}{3}\Big(\frac{\Delta x \Delta y}{\Delta x + \Delta y}\Big)\Big[\Big(\frac{q_A}{2\Delta y} + \frac{q_C}{\Delta y}\Big) + \Big(\frac{q_B}{\Delta x} + \frac{q_D}{2\Delta x}\Big)\Big] \\
&= \frac{2}{3}\Big[\Big(\frac{\Delta x}{\Delta x + \Delta y}\Big)(1/2\, q_A + q_C) + \Big(\frac{\Delta y}{\Delta x + \Delta y}\Big)(q_B + 1/2\, q_D)\Big]
\end{aligned}
\tag{3.18}
$$

For $\Delta x = \Delta y$,

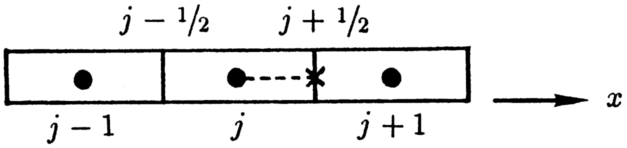
$$
q_E = 1/2\big[\underbrace{(1/3\, q_A + 2/3\, q_C)}_{\substack{\text{linear}\\ \text{interpolation}\\ \text{across } A-C}} + \underbrace{(2/3\, q_B + 1/3\, q_D)}_{\substack{\text{linear}\\ \text{interpolation}\\ \text{across B}-\text{D}}}\big]
\tag{3.19}
$$

3.4 Analysis of Reconstruction Scheme

Further insight into the character of the present reconstruction scheme can be gained through additional analysis. The scheme will be applied in the following to a one-dimensional stencil, a two-dimensional Cartesian grid, and a uniform triangular grid.

3.4.1 Uniform One-Dimensional Stencil

Consider a 1-D stencil with even spacing as shown in sketch (e):



Sketch (e)

Expand the solution in a Taylor series about $x = x_j$ to face $j + 1/2$

$$
q^{(-)}_{j+1/2} = q_j + \frac{dq}{dx}\Big|_j \Big(\frac{\Delta x}{2}\Big) + O(\Delta x^2)
\tag{3.20}
$$

Compute dq/dx using the nodal averages from Eq. (3.16) at $j + 1/2$ and $j - 1/2$ in one dimension:

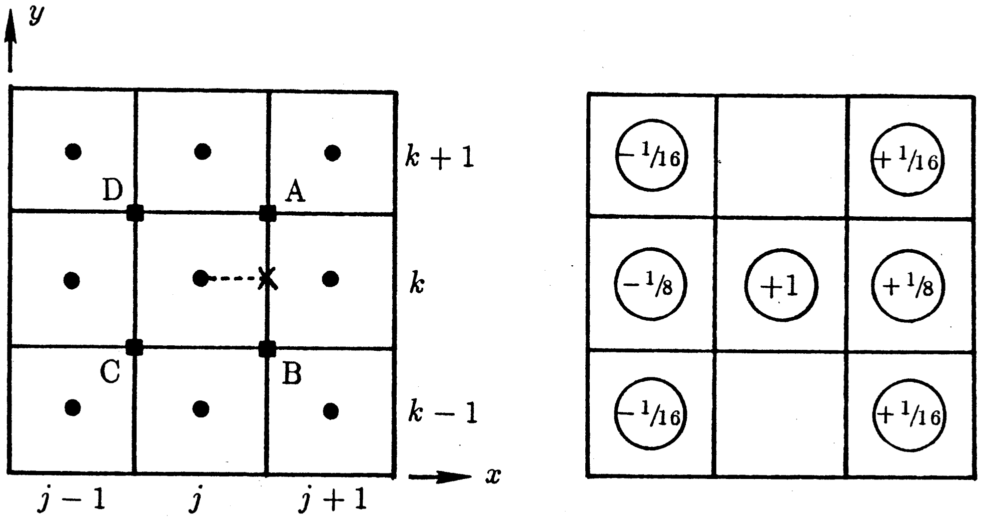
$$
\begin{aligned}
\frac{dq}{dx} &= \frac{q_{j+1/2} - q_{j-1/2}}{\Delta x} \\
&= \frac{1}{\Delta x}\Big[\Big(\frac{q_j}{\Delta x/2} + \frac{q_{j+1}}{\Delta x/2}\Big)\Big/\Big(\frac{1}{\Delta x/2} + \frac{1}{\Delta x/2}\Big) \\
&\quad - \Big(\frac{q_j}{\Delta x/2} + \frac{q_{j-1}}{\Delta x/2}\Big)\Big/\Big(\frac{1}{\Delta x/2} + \frac{1}{\Delta x/2}\Big)\Big] \\
&= \frac{q_{j+1} - q_{j-1}}{2\Delta x}
\end{aligned}
\tag{3.21}
$$

Substituting Eq. (3.21) into Eq. (3.20) results in Fromms' scheme [78]:

$$
\begin{aligned}
q^{(-)}_{j+1/2} &= q_j + 1/4(q_{j+1} - q_{j-1}) + O(\Delta x^2) \\
&= 1/4(q_{j+1} + 4q_j - q_{j-1}) + O(\Delta x^2).
\end{aligned}
\tag{3.22}
$$

3.4.2 Two-Dimensional Cartesian Grid

Consider 2-D Cartesian grid shown below in sketch (f):



Sketch (f)

Expand the solution about cell-centroid point j, k to cell face $j + 1/2, k$ and evaluate gradient

$$
\begin{aligned}
q^{(-)}_{j+1/2,k} &= q_{j,k} + \frac{\partial q}{\partial x}\Big(\frac{\Delta x}{2}\Big) + O(\Delta x^2) \\
&\approx q_{j,k} + \Big[\frac{1/2(q_A + q_B) - 1/2(q_C + q_D)}{\Delta x}\Big]\Big(\frac{\Delta x}{2}\Big) \\
&= q_{j,k} + 1/4(q_A + q_B - q_C - q_D)
\end{aligned}
\tag{3.23}
$$

Noting that all distances from the nodes to the cell centroids are equal, Eq. (3.16) yields

$$\begin{aligned} q_A &= {}^1\!/_4(q_{j,k} + q_{j,k+1} + q_{j+1,k+1} + q_{j+1,k}) \\ q_B &= {}^1\!/_4(q_{j,k} + q_{j,k-1} + q_{j+1,k-1} + q_{j+1,k}) \\ q_C &= {}^1\!/_4(q_{j,k} + q_{j,k-1} + q_{j-1,k-1} + q_{j-1,k}) \\ q_D &= {}^1\!/_4(q_{j,k} + q_{j,k+1} + q_{j-1,k+1} + q_{j-1,k}) \end{aligned} \tag{3.24}$$

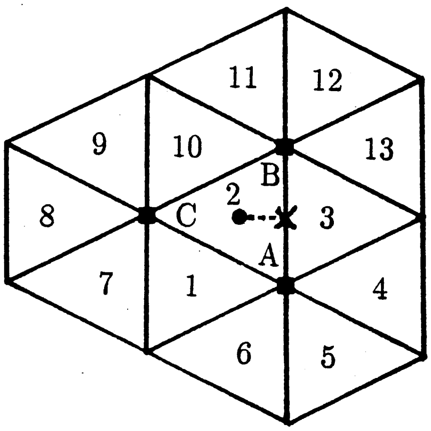
Substituting Eq. (3.24) into Eq. (3.23) gives

$$q^{(-)}_{j+1/2,k} = q_{j,k} + {}^1\!/_{16}[(q_{j+1,k+1} + 2q_{j+1,k} + q_{j+1,k-1}) - (q_{j-1,k+1} + 2q_{j-1,k} + q_{j-1,k-1})] \tag{3.25}$$

As illustrated to the right in sketch (f), Eq. (3.25) produces a 7-point stencil for the cell reconstruction.

3.4.3 Uniform Triangular Grid

The reconstruction formula for two-dimensional triangular grid has been derived in Appendix A2 as Eq. (A2.13). Consider a uniform triangular grid consisting of equilateral triangles:

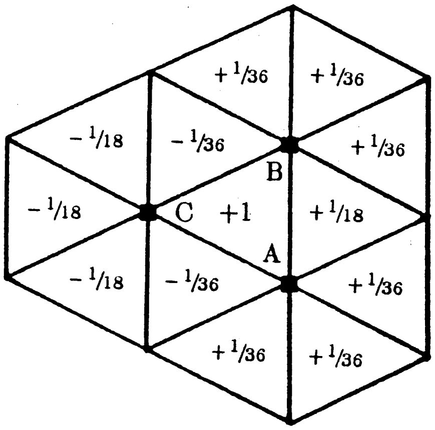


Sketch (g)

Note that all distances from the cell centroid to the nodes are equal, thus Eq. (3.16) simply produces an arithmetic average of the states from the cells surrounding a node. Applying Eq. (A2.13) to expand the solution from the centroid of cell 2 to edge A-B:

$$
\begin{aligned}
q_{AB} &= q_2 + {}^1\!/\!_3[{}^1\!/\!_2(q_A + q_B) - q_C] \\
&= q_2 + ({}^1\!/\!_6)({}^1\!/\!_3)[{}^1\!/\!_2(q_1 + 2q_2 + 2q_3 + q_4 + q_5 + q_6 + q_{10} + q_{11} + q_{12} + q_{13}) \\
&\qquad\qquad\qquad\qquad - (q_1 + q_2 + q_7 + q_8 + q_9 + q_{10})] \\
&= q_2 + {}^1\!/\!_{36}[\ 2(q_3 - q_7 - q_8 - q_9) + (q_4 + q_5 + q_6 - q_1 + q_{11} + q_{12} + q_{13} - q_{10})\ ]
\end{aligned}
\tag{3.26}
$$

Sketch (h) provides a visual depiction of the reconstruction stencil corresponding to Eq. (3.26).



Sketch (h)

As can be observed, information is used from all cells surrounding the candidate cell, but is biased across edge A-B.

3.5 Discussion of Limiting

Upwind schemes generally require the use of limiter functions to obtain smooth higher-order solutions around flow discontinuities, which has been the experience with the schemes of Refs. [38-42]. While the present method has yet to be applied in two-dimensions, experience with its application in three-dimensions has shown that limiting is not required, and that the method correctly captures shocks without oscillations. While this unexpected result is beneficial, it seems unlikely that a high-order scheme could capture discontinuities smoothly without some form of artificial dissipation being added. It is quite possible that the application of the averaging procedure across discontinuities could introduce additional dissipation and a local reduction in accuracy as is characteristic of limiters. This result may also be attributed in part to the use of the method in three dimensions and therefore on relatively coarse meshes. Application of the method in two dimensions on a sufficiently fine mesh may require some form of limiting to eliminate oscillations. With respect to limiting overshoots in the expansion of Eq. (3.9), it can be reasoned that when the averaging procedure of Eq. (3.16) is applied at a node, the resulting $\mathbf{q}_n$ represents a weighted mean value of the surrounding solution, i.e. $\mathbf{q}_n$ is bounded by the extrema of the surrounding solution. Furthermore, in three dimensions the summation of Eq. (3.16) accesses an average of 20 to 22 cells for each node, which results in a smoothing of errors introduced from the surrounding solution. Thus, the expansion will have been smoothed and bounded by the procedure and should not introduce new extrema into the solution.

4. TIME INTEGRATION

4.1 Introduction

The governing equations can be advanced in time by one of two basic methods: explicit or implicit time integration. An *explicit* scheme is one for which only one unknown appears in the difference equation in a manner which permits evaluation in terms of known quantities. It has the advantage of being economical in memory, simple to program, and readily vectorizable. The primary disadvantage is a restriction on the time step dictated by the need for explicit schemes to satisfy the Courant, Friedrichs, and Lewy (CFL) convergence condition for hyperbolic equations. An *implicit* scheme requires the simultaneous solution of a system of equations involving all of the unknowns at the new time level. The system of equations can be derived by linearizing the nonlinear residual function $\mathbf{R}$. The resulting set of algebraic equations can be solved by either a direct method or an iterative technique. In general, implicit schemes permit larger time steps since they are usually unconditionally stable, but require more memory and more computation time per time step. The most popular explicit scheme is Runge-Kutta time stepping developed by Jameson [67] which has been used quite successfully in a number of unstructured flow solvers [28-31, 33-44]. Implicit schemes for unstructured methods are still an active area of research [32, 39, 41, and 66] and are not widely used at this time. The Runge-Kutta time stepping scheme will be employed in the present work.

4.2 Runge-Kutta Time Stepping

The finite-volume discretization procedure in §2.3 leads to a set of semidiscrete coupled ordinary differential equations which must be integrated in time to obtain the steady state solution. Thus, writing Eq. (2.6) as:

$$V_i \frac{\partial \mathbf{Q}_i}{\partial t} + \mathbf{R}_i = \mathbf{0}, \qquad i = 1, 2, 3, \ldots \tag{4.1a}$$

where

$$\mathbf{R}_i = \sum_{j=1}^{\kappa(i)} \mathbf{F}_{i,j} \Delta A_{i,j} \tag{4.1b}$$

R_i is the residual accrued by summation of the fluxes through the four faces κ of a tetrahedral cell i. These equations are integrated in time using a fully explicit m-stage Runge-Kutta time-stepping scheme [67]:

$$\begin{aligned}
\mathbf{Q}_i^{(0)} &= \mathbf{Q}_i^n \\
\mathbf{Q}_i^{(1)} &= \mathbf{Q}_i^{(0)} - \alpha_1 \frac{\Delta t}{V_i} \mathbf{R}_i^{(0)} \\
&\cdots \\
\mathbf{Q}_i^{(m-1)} &= \mathbf{Q}_i^{(0)} - \alpha_{m-1} \frac{\Delta t}{V_i} \mathbf{R}_i^{(m-2)} \\
\mathbf{Q}_i^{(m)} &= \mathbf{Q}_i^{(0)} - \alpha_m \frac{\Delta t}{V_i} \mathbf{R}_i^{(m-1)} \\
\mathbf{Q}_i^{n+1} &= \mathbf{Q}_i^{(m)}
\end{aligned} \tag{4.2}$$

where the superscript n denotes the time level, and the parenthetical superscripts the stage of the Runge-Kutta time stepping. The weighting factors α_1 to α_m are defined as:

$$\alpha_k = \frac{1}{m-k+1}, \qquad k = 1, ..., m$$

These values of α_k will give m-order accuracy in time for a linear equation. Preliminary calculations were made using both a 3-stage and 4-stage scheme which are only second-order accurate for nonlinear equations [79]. The solution and convergence characteristics were essentially identical. Thus, a 3-stage scheme was used for the calculations presented in this paper.

In many cases, time accuracy in the integration is not required. For such cases, the solution convergence to steady state is accelerated by local time stepping and implicit residual smoothing.

4.3 Convergence Acceleration

4.3.1 Local Time Stepping

Local time stepping accelerates convergence by advancing the solution at each cell in time at a CFL number near the local stability limit. The expression for the local time step was derived with the aid of a 2-D stability analysis presented in Ref. [40]:

$$\Delta t_i \leq (CFL)\frac{V_i}{B_i + C_i + D_i} \tag{4.3}$$

with

$$B_i = (\mid u_i \mid + a_i)A_i^{(x)}$$

$$C_i = (\mid v_i \mid + a_i)A_i^{(y)}$$

$$D_i = (\mid w_i \mid + a_i)A_i^{(z)}$$

where CFL is the Courant-Friedrichs-Lewy number, V_i is the cell volume, a_i is the local speed of sound, and $A_i^{(x)}, A_i^{(y)}$, and $A_i^{(z)}$ are the projected areas of cell i in the x, y, and z directions. A method for computing projected area terms in Eq. (4.3) is presented in Appendix A1. The local time steps were updated every 50 cycles for the results presented in this paper.

4.3.2 Implicit Residual Smoothing

The maximum time step can be further increased by increasing the support of the scheme through implicit averaging of the residuals [10] with their neighbors. The residuals are filtered through a smoothing operator (which is essentially the Laplacian operator for a uniform grid):

$$\overline{\mathbf{R}_i} = \mathbf{R}_i + \varepsilon \nabla^2 \overline{\mathbf{R}_i}$$

where

$$\nabla^2 \overline{\mathbf{R}_i} = \sum_{j=1}^{\kappa(i)} (\overline{\mathbf{R}_j} - \overline{\mathbf{R}_i})$$

which effectively builds in some implicitness into the explicit time integration scheme. The summation uses residuals from the neighboring cells which share the faces κ

with cell i. The resulting set of equations can easily be solved iteratively by using Jacobi iteration

$$\overline{\mathbf{R}_i}^{(m)} = (\mathbf{R}_i + \varepsilon \sum_{j=1}^{\kappa(i)} \overline{\mathbf{R}_j}^{(m-1)}) \;/\; (1 + \varepsilon \sum_{j=1}^{\kappa(i)} 1) \tag{4.4}$$

Residual smoothing was performed during every stage of the Runge-Kutta time cycle. With two Jacobi iterations and a value of 0.5 for ε, the local time step can be doubled. Section 7.1 presents the results of a parameter study for assessing the sensitivities of ε and the number of Jacobi iterations on convergence.

5. BOUNDARY CONDITIONS

5.1 Flow Tangency

For the solid boundaries such as the wing and centerplane, the flow tangency condition is imposed by setting the velocities on the boundary faces to their cell-centered values and then subtracting the component normal to the solid surface. Density and pressure boundary conditions are simply set to the cell-centered value. A condition of zero mass and energy flux through the surface is ensured by setting the left and right states of solid boundary faces equal to the boundary conditions prior to computing the fluxes with Roe's approximate Riemann solver. This technique only permits a flux of the pressure terms of the momentum equations through a solid boundary.

5.2 Far-Field

Characteristic boundary conditions are applied to the far-field subsonic boundary using the fixed and extrapolated Riemann invariants corresponding to the incoming and outgoing waves traveling in characteristic directions defined normal to the boundary [10, 57]. The two locally one-dimensional Riemann invariants are given by

$$R^{\pm} = U \pm \frac{2a}{\gamma - 1} \tag{5.1}$$

The incoming Riemann invariant R^- is determined from the freestream flow and the outgoing invariant R^+ is extrapolated from the interior domain. The invariants are used to determine the locally normal velocity component and speed of sound:

$$\begin{aligned} \overline{U} &= {}^1\!/\!_2(R^+ + R^-) \\ \overline{a} &= \frac{\gamma - 1}{4}(R^+ - R^-) \end{aligned} \tag{5.2}$$

Since cell faces are defined with exterior unit normals, the sign of $\overline{U}$ is positive for outflow and negative for inflow. At an outflow boundary where $\overline{U} > 0$, the two tangential velocity components are extrapolated from the interior with the result

$$\begin{aligned} u_b &= u_{int} + n_x(\overline{U} - U_{int}) \\ v_b &= v_{int} + n_y(\overline{U} - U_{int}) \\ w_b &= w_{int} + n_z(\overline{U} - U_{int}) \end{aligned} \tag{5.3a}$$

Similarly for an inflow boundary where $\overline{U} < 0$, extrapolating the two tangential components of the freestream flow gives

$$\begin{aligned} u_b &= u_\infty + n_x(\overline{U} - U_\infty) \\ v_b &= v_\infty + n_y(\overline{U} - U_\infty) \\ w_b &= w_\infty + n_z(\overline{U} - U_\infty) \end{aligned} \tag{5.3b}$$

The density boundary condition is computed from the entropy relation

$$\rho_b = \left(\frac{\overline{a}^2}{\gamma S} \right)^{\frac{1}{\gamma - 1}} \tag{5.4}$$

where the entropy $S = S_{int}$ for an outflow boundary ($\overline{U} > 0$) and $S = S_\infty$ for an inflow boundary ($\overline{U} < 0$). The pressure condition is computed from the equation of state

$$p_b = \frac{\rho_b \overline{a}^2}{\gamma} . \tag{5.5}$$

When solving for the boundary fluxes, the boundary conditions are stored in an array and used in defining the right state in Eq. (3.1).

6. PROGRAMMING STRATEGY

6.1 Introduction

Historically, unstructured flow solvers utilize significantly more computer resources than do structured codes to solve a particular problem. Three-dimensional structured Euler codes generally require memory in the range of 40 to 50 words/cell and CPU times in the range of 25 to 35 μs/cell/cycle on a CRAY-2S class supercomputer. The efficiencies of seven three-dimensional unstructured flow solvers were assessed during a workshop entitled "Accuracy of Unstructured Grid Techniques" held at NASA Langley Research Center on January 16-17, 1990. The comparisons revealed memory requirements ranging from 73 to 306 words/(cell or node)[†] and run times efficiencies from 64 to 222 μs/(cell or node)/cycle. An earlier version of the present code [43] was one of the seven codes and required 73 words/cell and 65 μs/cell/cycle. It is apparent from the workshop results that careful attention should be given to both memory management and vectorization issues if unstructured codes are to become competitive with structured codes. Thus, the programming strategy itself becomes an issue in the effectiveness of unstructured flow solvers. These issues will be addressed in detail in the following.

6.2 Memory Management

All coding for the present work is written in standard FORTRAN 77 programming language. The memory is allocated within two single-dimensioned arrays for real and integer variables. The allocation of memory within the single-dimensioned arrays is controlled by pointers. A precise allocation of the memory is easily controlled by a parameter statement in the main program which is set once the problem size, i.e. number of cells, nodes, etc. is known after the grid has been generated:

[†] To avoid possible confusion, it should be noted that a 3-D structured mesh of hexahedral elements contains the same number of nodes as cells (asymptotically), whereas an unstructured mesh of tetrahedral elements generally contains between 5 and 6 times more cells than nodes. Thus, it is important to base efficiency comparisons on the number of unknowns computed, i.e. the number of cells for a cell-centered scheme and number of nodes for a node-centered scheme.

```
      PARAMETER (NCDIM = 108755,  !  TOTAL NUMBER OF CELLS
     .           NNDIM =  20412,  !  TOTAL NUMBER OF NODES
     .           NBNDIM=   4931,  !  NUMBER OF BOUNDARY NODES
     .           NFDIM = 222439,  !  TOTAL NUMBER OF CELL FACES
     .           NBFDIM=   9858)  !  NUMBER OF BOUNDARY FACES
C............................................................
C WORK SPACE . . .
      PARAMETER (MWORK = 5*NFDIM+5*NNDIM+10*NCDIM)
C
C REAL ARRAYS . . .
      PARAMETER (NWDIM =   4*NFDIM  !\
     .                 + 6*NCDIM  ! \
     .                 + 1*NNDIM  !  >   Overhead (See Fig. 2a)
     .                 + 9*NBFDIM ! /
     .                 +   MWORK) !/
C
C INTEGER ARRAYS . . .
      PARAMETER (IMWORK=   7*NFDIM
     .                 +  1*NBFDIM
     .                 + 13*NCDIM
     .                 +  2*NNDIM)
C
      DIMENSION W(NWDIM), IW(IMWORK)
```

The real array “W” is divided into two parts, one for “overhead” parameters which must be retained throughout the entire program execution, and the other for dynamic work space. The work space is dynamic with memory locations continually being used and overwritten. Figure 2 illustrates the utilization of the array space. In Fig. 2b, the diagram illustrates the relative positions of the intermediate variables in the work array as they are overwritten. Each column in Fig. 2b depicts what nominally occupies the work space at different steps of the iteration process. Note that the entire flux vector, FLUX, is stored prior to summation of the residuals. As will be explained in the next section, the storage of this vector leads to a straight forward vectorization of the algorithm which results in improved computational speed. This step does not increase the overall memory requirement since the same memory locations are used during the implicit residual smoothing procedure.

6.3 Vectorization

The general approach to vectorization is to construct the "do loops" so that the primary index of any arrayed variable to the left of an equal sign is identical to that of the loop index. This enables the FORTRAN compiler on Cray Research, Inc. supercomputers to vectorize the loops automatically. With this approach, the fluxes are computed and stored during a single pass over all of the cell faces as shown in the following example:

```
C..Compute averaged "Q" at node points   (See Section 3.3.3)
      CALL QNODE(... QN ...)
C
      DO 1 I=1,NFBND   !                    !  Loop over boundary faces
      NC1=IFACE(I,1)   ! Cell No. 1                    \
      MBC=IFACE(I,2)   ! Boundary Condition on face "I"  \
      ND1=IFACE(I,3)   ! Node No. 1 on face "I"           \   See
      ND2=IFACE(I,4)   ! Node No. 2 on face "I"           /   Fig. 3
      ND3=IFACE(I,5)   ! Node No. 3 on face "I"          /
      ND4=NDFACE(I,1)  ! Fourth node on cell NC1        /
      Q1 = FUNCTION("Q" AT NC1, AND "QN" AT ND1,ND2,ND3,ND4)
      IF(MBC FOR SOLID SURFACE) THEN  !            \       Insure no mass
      Q1 = QB(I)  ! Set Q1 to Boundary Condition    \      flux across
      ENDIF       !                                 /      solid boundary
      Q2 = QB(I)  ! Set Q2 to Boundary Condition   /
      FLUX(I) = FUNCTION(Q1,Q2,METRICS)             !  Roe's FDS scheme
    1 CONTINUE
```

```
C
        NFB1 = NFBND + 1
        DO 2 I=NFB1,NFACE              !  Loop over interior faces
        NC1=IFACE(I,1)   ! Cell No. 1               \
        NC2=IFACE(I,2)   ! Cell No. 2                \
        ND1=IFACE(I,3)   ! Node No. 1 on face "I"     \    See
        ND2=IFACE(I,4)   ! Node No. 2 on face "I"      >   Fig.  3
        ND3=IFACE(I,5)   ! Node No. 3 on face "I"     /
        ND4=NDFACE(I,1)  ! Fourth node on cell NC1   /
        MD4=NDFACE(I,2)  ! Fourth node on cell NC2  /
        Q1 = FUNCTION("Q" AT NC1, AND "QN" AT ND1,ND2,ND3,ND4)
        Q2 = FUNCTION("Q" AT NC2, AND "QN" AT ND1,ND2,ND3,MD4)
        FLUX(I) = FUNCTION(Q1,Q2,METRICS)             !  Roe's FDS scheme
      2 CONTINUE
```

This approach requires additional connectivity arrays IFACE and NDFACE. The IFACE array, defined in the example, is generated and stored by a simple preprocessor code which also assigns the boundary condition types to the appropriate boundary faces. Note that the first NFBND array locations of IFACE correspond to boundary faces. The boundary condition types are stored in locations IFACE(1-NFBND, 2). The array NDFACE, also defined in the preceding example, is generated during initial program execution. Higher-order estimates of the left and right states Q1 and Q2 shown in the example are obtained from Eq. (3.9). The fluxes are computed from Eq. (3.1).

The residuals (Eq. 4.1b) are accrued in a similar manner by looping once over the cells to sum the fluxes across the four tetrahedral cell faces. Again, this is illustrated by an example:

```
      DO 3 NC=1,NCELL
      MF1    =ICFACE(NC,1)    !  \
      MF2    =ICFACE(NC,2)    !   \   Sign-coded face numbers
      MF3    =ICFACE(NC,3)    !   /   for tetrahedral cell NC
      MF4    =ICFACE(NC,4)    !  /
      NF1    =IABS(MF1)
      NF2    =IABS(MF2)
      NF3    =IABS(MF3)
      NF4    =IABS(MF4)
      SIGN1  =FLOAT(MF1/NF1)  !  \
      SIGN2  =FLOAT(MF2/NF2)  !   \   = +1.  if NC=NC1 in IFACE
      SIGN3  =FLOAT(MF3/NF3)  !   /   = -1.  if NC=NC2 in IFACE
      SIGN4  =FLOAT(MF4/NF4)  !  /
      RES(NC)=SIGN1*FLUX(NF1)
     .       +SIGN2*FLUX(NF2)
     .       +SIGN3*FLUX(NF3)
     .       +SIGN4*FLUX(NF4)
    3 CONTINUE
```

This procedure requires the additional array ICFACE to be generated during startup which contains a list of the four faces of each tetrahedral cell. The signs on the four face indices control whether flux is added or subtracted from the residual of a cell. The sign convention that is established during the generation of the metric terms is to add the flux if NC=NC1 in the IFACE array, and subtract if NC=NC2. As can be observed, the algorithm maintains the conservative property of the finite volume formulation.

The present code requires 64 words/cell in memory and runs at a speed of 34 μs/cell/cycle on a single processor of a CRAY-2S with three stages of Runge-Kutta time stepping and two Jacobi iterations for implicit residual smoothing. One disadvantage of the present algorithm is the need to store the additional array ICFACE with a penalty of 4*NCELL in memory. An alternate data structure which requires slightly less memory is used in the methods of Ref. [28, 37, and 40]. The referenced algorithms do not store the fluxes, but perform the entire operation contained in the preceding two examples within a single loop over the faces. While those algorithms retain the desired property that the fluxes need only be computed once for each face, they do contain recursions in the residual summation which complicate vectorization. The recursions are reduced by reordering the faces, thereby enhancing vectorization. This alternate approach was not investigated in the present work.

7. COMPUTATIONAL RESULTS

A range of results are presented in this section to show the speed, accuracy and robustness of the new flow solver. The speed and the accuracy is shown by way of grid sensitivity and parameter studies while its robustness is established by flow solutions on several complex three-dimensional configurations. All the meshes for this study were generated using an improved version of the advancing front grid generation program, VGRID3D [51].

7.1 ONERA M6 Wing

The ONERA M6 wing has been used widely as a benchmark case to evaluate performance of newly developed flow solution methods. The wing has a leading edge sweep of 30 degrees, an aspect ratio of 3.8, a taper ratio of 0.56, and symmetrical airfoil sections. The wing has a root chord of 0.67 and a semispan b of 1.0 with a rounded tip. The computational domain is bounded by a rectangular box with boundaries at $-6.5 \leq x \leq 11.0$, $0.0 \leq y \leq 2.5$, and $-6.5 \leq z \leq 6.5$.

7.1.1 Grid Sensitivity

Transonic solutions were computed on three grids (Figs. 4 and 5) at the same conditions: $M_\infty = 0.84$, and $\alpha = 3.06°$, to make an assessment of the grid sensitivity. One of the meshes (Mesh 1) has cells stretched in the spanwise direction where gradients are small while the other two meshes have no stretching. Mesh size specifications are listed in Table 2.

Table 2. Mesh size specifications.

	Mesh 1	Mesh 2	Mesh 3
Total Cells	35008	108755	231507
Boundary Faces	4046	9858	16984
Total Nodes	6910	20412	42410
Boundary Nodes	2025	4931	8494

The computations were performed using the 3-stage Runge-Kutta time stepping scheme with local time stepping, implicit residual smoothing, and a CFL number of 4.0. Two cycles of the implicit residual smoothing were performed during each Runge-Kutta stage with $\varepsilon = 0.5$. The solutions were started from freestream initial conditions with the first-order scheme and run until the L_2-norm (RMS average of all residuals) decreased one order of magnitude, at which time the solver automatically switched to the higher-order scheme. The solution history is plotted in Fig. 6 against both CRAY-2S CPU time and number of cycles to provide a relative comparison of computational effort. Figure 6(a,b) shows the L_2-norm with a decrease of approximately 2.5 orders of magnitude. The convergence history of the lift coefficient is shown in Fig. 6(c,d). Additional details of the solution characteristics are provided in Table 3.

Table 3. Solution characteristics.

	Dimensioned Memory, mw	CRAY-2S Time, min.	Number Cycles
Mesh 1	2.3	12	660
Mesh 2	7.0	89	1565
Mesh 3	14.9	203	1716

A comparison of wing surface pressure contours for the three meshes is presented in Fig. 7 with contour intervals of $\Delta(p/p_\infty) = 0.02$. Before plotting, the computed face-centered boundary quantities were averaged to the boundary nodes using Eq. (3.16). The contour results show very little overall sensitivity to mesh size. As expected, the primary effect of the grid occurs with the spatial resolution of the shocks.

Pressure contours at the plane of symmetry are presented in Fig. 8 and again shows little sensitivity to mesh size. The primary differences occur with the resolution of the aft shock in the midchord region where the larger cells (see Fig. 5) limit spatial resolution.

Figure 9 shows the effect of mesh size on the streamwise surface C_p distribution at six span stations. Both the present unstructured results and inviscid solutions from the structured upwind code, CFL3D [80], are plotted in comparison to experimental data at a Reynolds number of 11.7 million [81], corresponding to conditions for which viscous effects are relatively small. The computations on the three tetrahedral meshes agree well with experiment and demonstrate comparable accuracy with the structured-grid calculations. The primary effect of mesh size is confined to regions of large gradients such as the leading-edge suction peak and the shock, where the finer mesh yields sharper shock definition. Although the deterioration of the solution for Mesh 1 is greater at the root and tip stations, it should be noted that its solution was obtained with an order of magnitude less CPU time and over 6 times less memory than that for Mesh 3.

The force and moment coefficients listed in Table 4 were computed by integrating the face-centered boundary pressures. The coefficients for lift, drag, pitching moment, and wing root bending moment are based on reference quantities of $A_{ref} = .5255$, $\bar{c} = .67$, and $b_{ref} = 1.0$. The pitching moment is referenced about the wing apex.

Table 4. Force and moment characteristics.

	Mesh 1	Mesh 2	Mesh 3
C_L	.2816	.2904	.2911
C_D	.0141	.0132	.0123
C_m	-.1688	-.1724	-.1726
C_{RBM}	.1270	.1283	.1285

The chordwise entropy distributions presented in Fig. 10 are defined by the relation:

$$\text{Entropy Parameter} = \frac{\gamma p}{\rho^\gamma} - 1 \tag{7.1}$$

For these supercritical inviscid solutions, the flow should be isentropic ahead of the shocks with zero entropy production. Entropy should only be produced through the shock waves and convected downstream thereafter. Any generation of entropy apart from shocks or even an unphysical reduction of entropy can be only interpreted as numerical error. Figure 10 shows the presence of erroneous entropy peaks at the leading edge above a level of 0.03 for Meshes 1 and 2. Refinement of the mesh (Mesh 3) results in a substantial reduction in these entropy peaks. The finer Mesh 3 also produces a further decrease of the entropy convected downstream of the erroneous peaks. Further reductions in the peaks may be possible by additional mesh refinement or by applying alternate boundary conditions which maintain a zero gradient of entropy normal to the solid surface [82]. The entropy rise through the shocks generally overshoots but settles to a constant value.

7.1.2 Implicit Residual Smoothing

A study is performed to investigate the effect of the IRS parameters in Eq. (4.4) on solution convergence. Convergence assessments are made for the L_2-norm and lift coefficient in terms of both CPU time and number of iterations. A further study to determine the impact of implicit residual smoothing on the maximum CFL number is also performed.

Figure 11 shows the effect of the number of Jacobi iterations, JITER, applied to Eq. (4.4) for Mesh 2. The Jacobi iterations were varied from 2 to 4 using fixed values for $\varepsilon=0.5$ and CFL=4. The solution for this case diverged with JITER=1. Shown for reference is the case with no IRS (JITER=0). There, the upper limit for stable convergence is reached with CFL$\approx$ 2. It is evident in Figs. 11(b) and 11(d) that implicit residual smoothing significantly reduces the total number of cycles required for convergence. However, the procedure does impose additional overhead which is reflected in the results shown in Figs. 11(a) and 11(c). These latter two figures demonstrate that IRS is beneficial in substantially reducing the total CPU time required for convergence with two Jacobi iterations. No significant advantage is gained by using more than two Jacobi iterations for the fixed values of ε and CFL.

The effect of varying the coefficient ε with JITER=2 and CFL=4 is shown in Fig. 12. A decrease in ε results in more rapid convergence of the L_2-norm, but has little effect on the lift coefficient. Convergence could not be achieved with ε=0.1 or 0.9.

The benefit derived from using IRS to increase the numerical stability limit is illustrated in Fig. 13. With JITER=3, the CFL number can be increased up to a value of 6 for this wing. (With two Jacobi cycles, the CFL number could not be increased beyond a value of 4.) Once again, the case with no IRS (JITER=0) is shown for reference. Figure 13 shows a dramatic improvement in convergence rate with the higher CFL numbers. While these results are quite encouraging, experience gained by computing several complex configurations has shown that the best overall values for the IRS parameters are JITER=2, ε=0.5, and CFL=4. In some cases, it may be necessary to lower the CFL number further.

7.2 Boeing 747-200

Computations were performed on the Boeing 747-200 configuration to demonstrate the robustness of the flow solver for computing the flow around realistic transport configurations. As shown in Fig. 14, this configuration has two double-ringed flow-through nacelles and an empennage. The computational semispan grid has 216,860 cells and 39,868 total nodes. The surface grid contains 14,268 boundary faces and 7,128 boundary nodes, including the outer boundary and plane of symmetry. A solution was computed for $M_\infty = 0.84$ and $\alpha = 2.73°$ using CFL=4 and the convergence acceleration techniques. A wing reference area of 5,500 ft^2 was used for computing the lift coefficient.

Figure 15 portrays a good resolution of the surface pressure contours over the entire configuration. Note the presence of a double lambda shock on the wing.

The solution history is presented in figure 16. The solution was started with the first order scheme until the residual dropped one order of magnitude, then continued with the higher-order scheme for a total of 2000 cycles. The L_2-norm decreases 2.7

orders of magnitude in approximately 190 minutes of CRAY-2S CPU time. The lift coefficient has converged well before that time. The solution was started from freestream initial conditions and required 13.9 megawords of memory.

7.3 Low-Wing Transport

Another quantitative assessment of the flow solver is made using a low-wing transport (LWT) configuration described in Ref. [83]. The 1/17th-scale configuration contains a supercritical airfoil and a flow-through representation of an advanced turbofan nacelle with a bypass ratio of approximately 6. The experimental pressure measurements were obtained in the NASA Langley 16-Foot Transonic tunnel [84] at transonic Mach numbers with Reynolds numbers in the range of 2.5 x 10^6 based on the mean aerodynamic chord of the wing. The present calculations were made for the condition of $M_\infty = 0.768$ and $\alpha = 1.116°$.

The computational grid consists of 418,939 cells and 75,470 total nodes representing the semispan configuration. The surface grid (Fig. 17) contains 21,428 boundary faces and 10,716 boundary nodes on the semispan, including the outer boundaries and plane of symmetry. A sufficient definition of the internal flow-through nacelle geometry was not available, so the grid was terminated at a plane inside the inlet, and at the two bypass exit planes. Freestream conditions were prescribed on the exit plane boundaries. A condition of $M = 0.632 M_\infty$ was imposed on the inlet plane to balance the mass flux.

Surface pressure contours are presented in Fig. 18. Good resolution of the wing shock can be observed along with evidence of an inboard lambda shock.

A comparison of the streamwise C_p distributions are shown in Fig. 19 for six span stations. The inviscid results are compared with experimental data at one degree higher angle of attack. In general, the agreement is good and consistent with the expected effects of viscosity. The shock is more aft, and the effects of flow separation downstream of the shock and in the lower-surface cusp region result in a lower experimental ΔC_p over the aft region. (It should be noted that the stations within

$0.463 \leq \eta \leq .70$ had only three cells defining the region between the shock and the trailing edge, and thus do not adequately resolve the flow aft of the shock.) Similar comparisons are presented in Ref. [85] for a typical transport configuration with a supercritical wing. Comparisons shown between viscous and inviscid calculations and transonic experimental data show viscous effects comparable in magnitude to those observed in the present calculations.

The solution was obtained using mesh sequencing [44] and required approximately 6 hours of CRAY-2S CPU time and 27 megawords of memory. The present grid has a considerable number of cells clustered in directions of small gradients where they are not needed. The total number of cells could be greatly reduced with grid stretching, which would significantly decrease the amount of memory and CPU time required to obtain a satisfactory solution. Generation of stretched grids for complex configurations is presently an active area of research.

7.4 High-Speed Civil Transport

A generic High-Speed Civil Transport (HSCT) configuration was chosen because it represents a different class of geometry and flow field. This configuration is characterized by its slender planform which can generate leading-edge vortex flow under certain flow conditions. While the wind tunnel model [86] has a small radius along the leading edge, recent tests conducted by Pamela Belton in the NASA Langley 8-Foot Transonic Pressure Tunnel [87] revealed the presence of leading-edge vortex flow at relatively low angles of attack for transonic Mach numbers. Thus, a computational grid was constructed which was sufficiently coarse to provide a numerically sharp leading-edge, thereby inducing flow separation. The resulting surface grid, depicted in Fig. 20, contains 10,016 longitudinally stretched boundary faces and 5010 boundary nodes on the semispan, as well as the plane of symmetry and outer boundary. The semispan volume grid contains 184,997 cells and 33,499 total nodes. A computation was made for the conditions $M_\infty = 0.901$ and $\alpha = 6.47°$.

Figure 21 portrays the surface "oilflow" lines as computed from the solution. The outward sweeping of these surface streamlines indicates the the presence of leading-edge vortex flow on both the inboard and outboard panels.

The solution history is presented in Fig. 22. The solution was started with the first order scheme until the residual dropped one order of magnitude, then continued with the higher-order scheme for a total of 1000 cycles. With CFL=4 and convergence acceleration, the residual decreases 2.5 orders of magnitude in 58 minutes on a CRAY-YMP, which is approximately 37-percent faster than the CRAY-2S. This calculation required 11.8 megawords of memory.

7.5 Space Transportation System

A computation was made on the Space Transportation System (STS) ascent configuration at $M_\infty = 1.05$, and $\alpha = -3.1°$ to demonstrate the robustness of the flow solver in obtaining a solution on a complex geometry with a complex flow field. The semispan grid, which includes the orbiter, external tank, and solid rocket boosters, consists of 108,538 cells and 21,562 total nodes. The surface grid shown in Fig. 23 is represented by 13,552 triangular faces and 6,780 nodes, including the outer computational boundaries and plane of symmetry. The computations were made with zero elevon deflection.

Figure 23 shows a composite picture of the surface triangulation and the corresponding pressure contours on the full configuration. The centerplane grid is shown in Fig. 24 along with the pressure contours in Fig. 25. The basic features of the flow (shocks, expansions etc.) have been well captured in the solution considering that only 3-4 layers of fine cells have been used close to the body.

As shown in Fig. 26, the solution was obtained with 1900 first-order cycles using a CFL number of 0.5, then 2400 additional cycles at higher-order with a CFL number of 1.0 for a total residual reduction of 3.2 orders of magnitude. The solution required 142 minutes of CPU time on a CRAY-YMP and used 7 megawords of memory.

The quality of the present results, which were computed on a relatively coarse grid with a relatively small amount of computer time, serves as a good demonstration of the overall flow solver capabilities.

8. SUMMARY AND CONCLUSIONS

The objective for the present work was to develop a fast and accurate upwind scheme for solving the three-dimensional Euler equations on unstructured tetrahedral meshes. The general approach was to utilize proven technologies when possible to reduce risk. Thus, a cell-centered finite-volume formulation was implemented for tetrahedral cells using Roe's flux-difference splitting to compute interface fluxes. Solutions were advanced in time using a 3-stage Jameson-style Runge-Kutta time stepping scheme with convergence accelerated to steady state by local time stepping and implicit residual smoothing. An important new development was presented for efficiently constructing higher-order differences within tetrahedral cells. The differences are formed by a novel cell-reconstruction process that is based on an analytical evaluation of solution gradients within tetrahedral cells rather than the cumbersome numerical procedures typically employed by prior methodologies. As a result, the total run time was reduced by one-half. Careful attention was also given to vectorization and memory management. The resulting code runs at a speed of 34 microseconds per cell per cycle on a CRAY-2S supercomputer and requires 64 words of memory per cell, which is significantly lower than most contemporary unstructured flow solvers. To put this accomplishment in perspective, three-dimensional structured Euler codes generally operate at speeds in the range of 25 to 35 microseconds per cell per cycle on the same computer and require memory in the range of 40 to 50 words per cell. To further emphasize the significance of this result, comparisons were shown between the present method and structured-grid upwind results to demonstrate that comparable accuracies are attainable with comparable numbers of computational cells. Thus, the new method offers a competitive alternative to structured-grid Euler codes in terms of accuracy and efficiency, but has the additional benefit of capitalizing on the increased geometrical flexibility available through tetrahedral cells and on generalized indexing schemes.

Results have been presented for a range of configurations at transonic Mach numbers to demonstrate the speed, accuracy, and robustness of the flow solver. The computational efficiency and accuracy has been illustrated by obtaining solutions on the ONERA M6 wing in 12 minutes of CRAY-2S run time with 2.3 megawords of memory, and by comparing to experimental data and structured-grid results from a prominent upwind code, CFL3D. A solution was generated for a Boeing 747-200 configuration with empennage and dual-ringed flow-through nacelles to qualitatively assess the utility of the method for computing the complex flow-field around a realistic aircraft configuration. A quantitative assessment has been presented for a low-wing transport configuration to demonstrate the accuracy and robustness of the flow solver on a complex geometry. The ability of the method to compute leading-edge vortex flow on a realistic slender configuration was demonstrated for a generic high-speed civil transport. A computation has been made on the full Space Transportation System at a transonic Mach number to demonstrate the robustness of the flow solver in obtaining a solution on a very complex geometry with a complex flow field.

The present effort concludes with a new scheme for efficiently solving the three-dimensional Euler equations around very complex geometries using tetrahedral meshes. The coding was constructed to serve as a modular base platform on which to build additional capabilities. It is recognized that the unstructured grid methodology will not be fully utilized until viscous effects can be included for full-scale aircraft simulations. Thus, current plans include the addition of viscous shear terms, turbulence model, and multigrid convergence acceleration. Additional work is also needed in the area of viscous grid generation. An effort is also planned to install an existing iterative design algorithm into the code which will permit the design of aircraft components to match prescribed target pressure distributions. The flow solver is compatible with an existing advancing-front grid generator and a graphics post-processing code, and is actively being used for conducting applied aerodynamic research.

BIBLIOGRAPHY

1. Compendium of Abstracts from the NASA Computational Fluid Dynamics Conference held at NASA Ames Research Center, March 12-14, 1991. Sponsered by the Aerodynamic Division, Office of Aeronautics Exploration and Technology (OAET), NASA Headquarters, Washington, D.C. 20546.
2. Boppe, C. W., "Computational Transonic Flow About Realistic Aircraft Configurations", AIAA Paper No. 78-104, January 1978.
3. Boppe, C. W. and Stern, M. A.,"Simulated Transonic Flows for Aircraft with Nacelles, Pylons, and Winglets", AIAA Paper 80-0130, Jan. 1980.
4. Boppe, C. W., "Transonic Flow Field Analysis for Wing-Fuselage Configurations", NASA CR-3243, May 1980.
5. Jameson, A. and Caughey, D. A., "A Finite Volume Method for Transonic Potential Flow Calculations", AIAA 3rd Computational Fluid Dynamics Conference, Albuquerque, NM, June, 1977.
6. Szema, K. Y. and Shankar, V., "Nonlinear Computation of wing-Body-Vertical Tail-Wake Flows at Low Supersonic Speed", AIAA Paper 84-0427, Jan. 1984.
7. Samant, S. S., Bussoletti, J. E., Johnson, F. T., Burkhart, R. H., Everson, B. L., Melvin, r. G., Young, D. P., Erickson, L. L., Madson, M. D. and Woo, A. C., "TRANAIR: A Computer Code for Transonic Analyses of Arbitrary Configurations", AIAA Paper 87-0034, 1987.
8. Boppe, C. W., "Aerodynamic Analysis for Aircraft With Nacelles, Pylons, and Winglets at Transonic Speeds", NASA CR-4066, April 1987.
9. Rizzi, A. and Eriksson, L. E., "Transfinite Mesh Generation and Damped Euler Equation Algorithm for Transonic Flow Around Wing-Body Configurations", AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, June 1981.
10. Jameson, A. and Baker, T. J., "Solution of the Euler Equations for Complex Configurations", AIAA Paper 83-1929, July 1983.
11. Obayashi, S. and Fujii, K., "Toward the Navier-Stokes Analysis of Transport Aircraft Configurations", AIAA Paper 87-0428, Jan. 1987.

12. Rai, M. M., " A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations", AIAA Paper 84-0164, Jan. 1984.

13. Hessenius, K. A. and Rai. M. M., "Three Dimensional, Conservative, Euler Computations Using Patched Grid Systems and Explicit Methods", AIAA Paper 86-1081, May 1986.

14. Walters, R. W., Thomas, J. L., and Switzer, G. F., "Aspects and Applications of Patched Grid Calculations", AIAA Paper 86-1063, May 1986.

15. Lombard, C. K. and Venkatapathy, E., "Implicit Boundary Treatment for Joined and Disjoint Patched Mesh Systems", AIAA Paper 85-1503, Aug. 1985.

16. Szema, K. Y., Chakravarthy, S. R., Riba, W. T., Byerly, J., and Dresser, H. S., "Multi-Zone Euler Marching Technique for flow over Single and Multi-Body Configurations", AIAA Paper 87-0592, Jan. 1987.

17. Steger, J. L., Dougherty, F. C., and Benek, J. A., "A Chimera Grid Scheme", in *Advances in Grid Generation*, K. N. Ghia and U. Chia, Eds., ASME FED-5, pp. 59-69, 1983.

18. Atta, E. H. and Vadyak, J., "Numerical Simulation of the Transonic Flowfield for Wing/Nacelle Configurations", AIAA Paper 84-2430, Oct. 1984.

19. Baker, T. J., Jameson, A., and Vermeland, R. E., "Three-Dimensional Euler Solutions with Grid Embedding", AIAA Paper 85-0121, Jan. 1985.

20. Holst, T. L., Gundy, K. L., Flores, J., Chaderjian, N. M., Kaynak, U., and Thomas, S. D., "Numerical Solution of Transonic Flows Using an Euler/Navier-Stokes zonal Approach", AIAA Paper 85-1640, July, 1985.

21. Fujii, K., "A Method to Increase the Accuracy of Vortical Flow Simulations", AIAA Paper 88-2562, June 1988.

22. Flores, J. and Chaderjian, N. M., "The Numerical Simulation of Transonic Separated Flow About The Complete F-16A", AIAA Paper 88-2506, June 1988.

23. Buning, P. G., Chiu, I. T., Obayashi, S., Risk, Y. M., and Steger, J. L., "Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent", AIAA Paper 88-4359, Oct. 1988.

24. Rizk, Y. M. and Gee, K., "Numerical Prediction of the Unsteady Flowfield Around the F-18 Aircraft At Large Incidence", AIAA Paper 91-0200, Jan. 1991.

25. Ghaffari, F., Luckring, J. M., Thomas, J. L., and Bates, B. L., "Navier-Stokes Solutions About the F/A-18 Forebody-Leading-Edge Extension Configuration", *Journal of Aircraft*, Vol. 27, No. 9, Sept. 1990, pp.737,748.

26. Schiff, L. B., Cummings, R. M., Sorenson, R. L., and Rizk, Y. M., "Numerical Simulation of High-Incidence Flow Over the F-18 Fuselage Forebody", AIAA Paper 89-0339, Jan. 1989.

27. Ghaffari, F., Bates, B., Luckring, J., and Thomas, J., "Transonic Navier-Stokes Solutions About a Complex High-Speed Accelerator Configuration", AIAA Paper 90-0430, Jan. 1990.

28. Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow Over A Complete Aircraft", AIAA Paper 86-0103, Jan. 1986.

29. Jameson, A. and Baker, T. J., "Improvements to the Aircraft Euler Method", AIAA Paper 87-0452, Jan. 1987.

30. Stoufflet, B., Periaux, J., Fezoui, F., and Dervieux, A., "Numerical Simulation of 3-D Hypersonic Euler Flows Around Space Vehicles Using Adapted Finite Elements", AIAA Paper 87-0560, Jan. 1987.

31. Peraire, J., Peiro, J., Formaggia, L., Morgan, K., and Zienkiewicz, O. C., "Finite Element Euler Computations in Three Dimensions", AIAA Paper 88-0032, Jan. 1988.

32. Hassan, O., Morgan, K., and Peraire, J., "An Implicit Finite Element Method for High Speed Flows", AIAA Paper 90-0402, Jan. 1990.

33. Löhner, R. and Baum, J. D., "Three-Dimensional Store Separation Using A Finite Element Solver and Adaptive Remeshing", AIAA Paper 91-0602, Jan. 1991.

34. Bristeau, M., Mallet, M., Periaux, J., and Roge, G., "Development of Finite Element Methods for Compressible Navier-Stokes Flow Simulations in Aerospace Design", AIAA Paper 90-0403, Jan. 1990.

35. Löhner, R. and Baum, J. D., "Numerical Simulation of Shock Interaction with Complex Geometry Three-Dimensional Structures Using a New Adaptive H-Refinement Scheme on Unstructured Grids", AIAA Paper 90-0700, Jan. 1990.
36. Marcum, D. L. and Agarwal, R. K., "A Three-Dimensional Finite Element Navier-Stokes Solver with k-ε Turbulence Model for Unstructured Grids", AIAA Paper 90-1652, June 1990.
37. Mavriplis, D. J., "Three-Dimensional Unstructured Multigrid for the Euler Equations", AIAA Paper 91-1549, June 1991.
38. Barth, T. J.; and Jespersen, D. C.: The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA Paper 89-0366, 1989.
39. Venkatakrishnan, V.; and Barth, T. J.: Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes. AIAA Paper 89-0364, 1989.
40. Whitaker, D. L.: Two-Dimensional Euler Computations on a Triangular Mesh Using an Upwind, Finite-Volume Scheme. Ph.D Dissertation, Virginia Polytechnic Institute and State University, 1988.
41. Batina, J. T.: Three-Dimensional Flux-Split Euler Schemes Involving Unstructured Dynamic Meshes. AIAA Paper 90-1649, 1990.
42. Wey, T. C.; and Li, C. P.: Numerical Simulation of Shuttle Ascent Transonic Flow Using An Unstructured Grid Approach. Presented at "Symposium on Computational Technology for Flight Vehicles", Washington, D.C. Nov. 5-7, 1990.
43. Frink, N. T.: Upwind Scheme for Solving the Euler Equations on Unstructured Tetrahedral Meshes. Presented at workshop on "Accuracy of Unstructured Grid Techniques" held at NASA Langley Research Center, Jan. 16-17, 1990. Also accepted for publication in *AIAA Journal.*
44. Frink, N. T., Parikh, P., and Pirzadeh, S., "A Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes", AIAA 91-0102, Jan. 1991.

45. Reddy, J. N., "An Introduction to the Finite Element Method", McGraw-Hill Book Company, ISBN 0-07-051346-5, 1984.

46. Löhner, R. and Parikh, P., "Three-Dimensional Grid Generation by the Advancing Front Method", *Int. J. Num. Meth. Fluids 8*, 1988, pp. 1135-1149.

47. Weatherill, N. P., "The Generation of Unstructured Grids Using Dirichlet Tessalations", *Princeton University MAE Report No. 1715*, July 1985.

48. Baker, T. J., "Three-Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", AIAA Paper 87-1124.

49. Mavriplis, D. J., "Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation", NASA CR 181699, Aug. 1988.

50. Nakahashi, K., "Optimum Spacing Control of the Marching Grid Generation", AIAA Paper 91-0103, Jan. 1991.

51. Parikh, P., Pirzadeh, S., and Löhner, R., "A Package for 3-D Unstructured Grid Generation, Finite-Element Flow Solutions, and Flow-Field Visualization", NASA CR-182090, Sept. 1990.

52. Liepmann, H. W. and Roshko, A., *Elements of Gasdynamics*, John Wiley & Sons, Inc., ISBN 0-471-53460-9, 1957.

53. Walters, R. W. and Thomas, J. L., "Advances in Upwind Relaxation Methods", in *State-of-the-Art Surveys on Computational Mechanics*, ed. A. K. Noor, ISBN 0-7918-0303-1, ASME Publications, 1988.

54. Newsome, R. W., Walters, R. W. and Thomas, J. L., "An Efficient Iteration Strategy for Upwind/Relaxation Solutions to the Thin-Layer Navier-Stokes Equations", AIAA Paper No. 87-1113-CP, 1987.

55. Thomas, J. L. and Walters, R. W., "Upwind Relaxation Algorithms for the Navier-Stokes Equations", AIAA 85-1501, July 1985.

56. Thomas, J. L., Van Leer, B., and Walters, R. W., "Implicit Flux-Split Schemes for the Euler Equations", AIAA Paper 85-1680, July 1985.

57. Anderson, W. K., Thomas, J. L., and Whitfield, D. L., "Three-Dimensional Multigrid Algorithms for the Flux-Split Euler Equations", NASA TP 2829, Nov. 1988.

58. Vatsa, V. N., "Accurate Numerical Solutions for Transonic Viscous Flow over Finite Wings", *Journal of Aircraft*, Vol. 24, No. 6, June 1987, pp. 377-385.

59. Bonhous, D. L. and Wornom, S. F., "Relative Efficiency and Accuracy of Two Navier-Stokes Codes for Simulating Attached Transonic Flow Over Wings", NASA TP 3061, Feb. 1991.

60. Maksymiuk, C. M., Swanson, R. C., and Pulliam, T. H., "A Comparison of Two Central Difference Schemes for Solving the Navier-Stokes Equations", NASA TM 102815, July 1990.

61. Beam, R. M. and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", AIAA Journal, Vol. 16, No. 4, April 1978, pp. 393-402.

62. Chakravarthy, S. R., Szema, K. Y., Goldberg, U. C., Gorski, J. J., and Osger, S., "Application of a New Class of High Accuracy TVD Schemes to the Navier-Stokes Equations", AIAA 85-0165, Jan. 1985.

63. Briley, W. R. and McDonald, H., "Solutions of the Multi-Dimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method", *Journal of Computational Physics*, Vol. 24, 1977, pp. 372-397.

64. Pulliam, T. H. and Steger, J. L., "On Implicit Finite-Difference Simulations of Three-Dimensional Flow", AIAA 78-10, Jan. 1978.

65. Osher, S. and Soloman, F., "Upwind Difference Schemes for Hyperbolic Systems of Conservation Laws", *Mathematics of Computation*, Vol. 38, 1982, pp. 339-374.

66. Walters, R. W., "Compressible Flow Algorithms on Structured/Unstructured Grids", invited lecture presented at the Ninth International Conference on Computing Methods in Applied Sciences and Engineering, Paris, France, January 29 - Feburary 2, 1990.

67. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes", AIAA Paper 81-1259, June 1981.

68. Mavriplis, D. and Jameson, A., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes", AIAA Paper 87-0353, Jan. 1987.

69. Steger, J. L. and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite-Difference Methods", *Journal of Computational Physics*, Vol. 40, 1981.

70. Van Leer, B., "Flux-vector Splitting for the Euler equations", in *Lecture Notes in Physics*, Vol. 170, ISBN 0-387-11948-5, Springer-Verlag, 1982.

71. Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", *Journal of Computational Physics*, Vol. 43, 1981.

72. Van Leer, B., Thomas, J. L., Roe, P. L. and Newsome, R. W., "A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations", AIAA Paper No. 87-1104-CP, 1987.

73. Roe, P. L., "Characteristic-Based Schemes for the Euler Equations", *Annual Review of Fluid Mechanics*, Vol. 18, 1986.

74. Roe, P. L., " 'Optimum' Upwind Advection On A Triangular Mesh", NASA CR 187459, Oct. 1990.

75. Rumsey, C. L. and van Leer, B., "A Grid-Independent Approximate Riemann Solver With Applications to the Euler and Navier-Stokes Equations", AIAA Paper 91-0239, Jan. 1991.

76. Dadone, A. and Grossman, B., "A Rotated Upwind Scheme for the Euler Equations", AIAA Paper No. 91-0635, Jan. 1991.

77. Roe, P. L. and Pike, J., "Efficient Construction and Utilization of Approximate Riemann Solutions", in *Computing Methods in Applied Sciences and Engineering VI*, eds. R. Glowinski and J.-L. Lions, ISBN 0-444-87597-2, Elsevier Science Publishing Co., 1984.

78. Fromm, J. E., "A Method for Reducing Dispersion in Convective Difference Schemes", *Journal of Computational Physics*, Vol. 3, pp. 176-189.

79. Swanson, R. C., and Turkel, E., "A Multistage Time-Stepping Scheme for the Navier-Stokes Equations", AIAA Paper No. 85-0035, Jan. 1985.

80. Anderson, W. K., Thomas, J. L. and Van Leer, B., "Comparison of Finite Volume Flux Vector Splittings for the Euler Equations", *AIAA Journal*, Vol. 24, No. 9, 1986.

81. Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA M6-Wing at Transonic Mach Number", AGARD Advisory Report 138, May 1979.

82. Barton, J. T. and Pulliam, T. H., "Airfoil Computations at High Angles of Attack, Inviscid and Viscous Phenomena", *AIAA Journal*, Vol. 24, May 1986, pp. 705-714.

83. Pendergraft, O. C., Re, R. J., and Kariya, T. T., "Nacelle/Pylon Interference Study on a 1/17th-Scale, Twin-Engine, Low-Wing Transport Model", AIAA Paper 89-2480, July 1989.

84. Corson, B. W., Jr., Runckel, J. F., and Igoe, W. B., "Calibration of the Langley 16-Foot Transonic Tunnel With Test Section Air Removal", NASA TR R-423, 1974.

85. Melson, N. D. and Streett, C. L., "Updated User's Guide for TAWFIVE With Multigrid", NASA TM-4109, 1989.

86. Robins, A. W., Dollyhigh, S. M., Beissner, F. L., Jr., Geiselhart, K., Martin, G. L., Shields, E. W., Swanson, E. E., Coen, P. G., and Morris, S. J., Jr., "Concept Development of a Mach 3.0 High-Speed Civil Transport", NASA TM-4058, 1988.

87. Schaefer, W. T., Jr., "Characteristics of Major Active Wind Tunnels at the Langley Research Center", NASA TM X-1130, 1965.

FIGURES

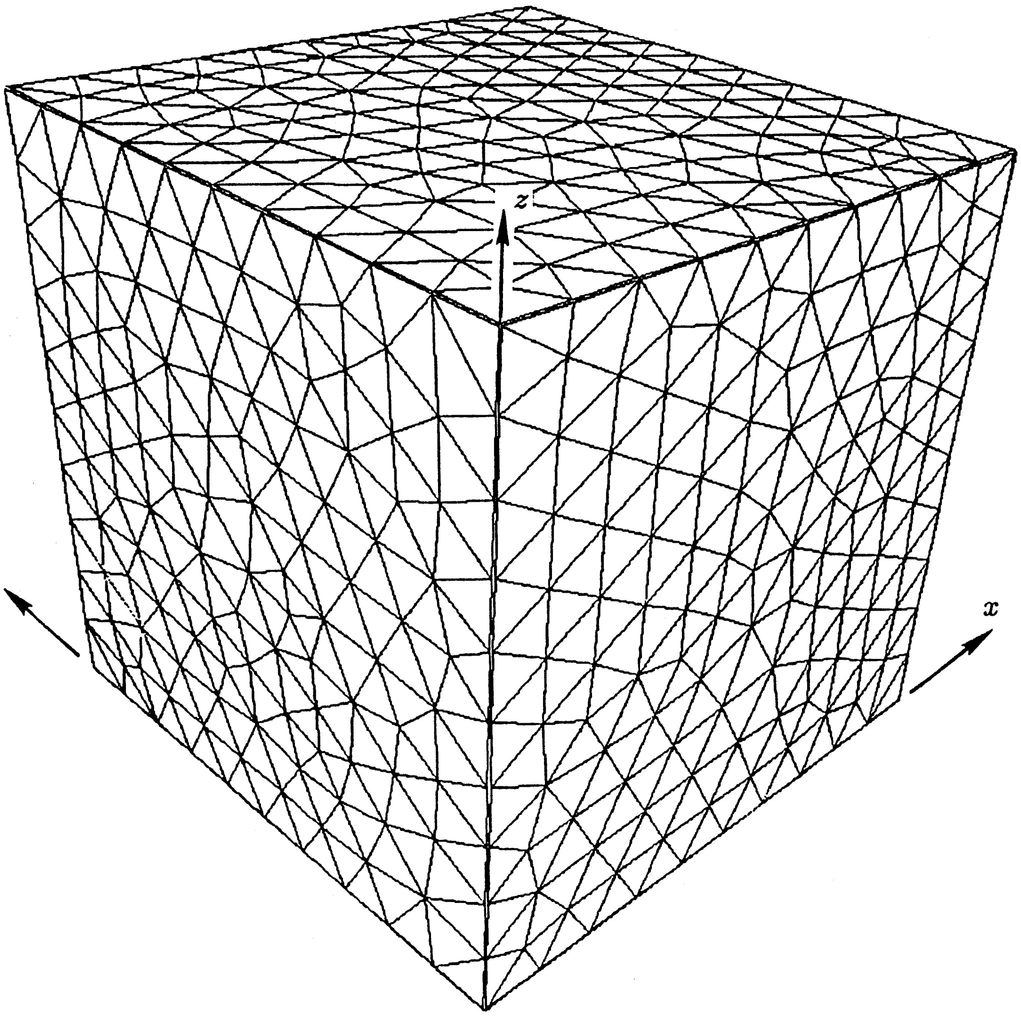
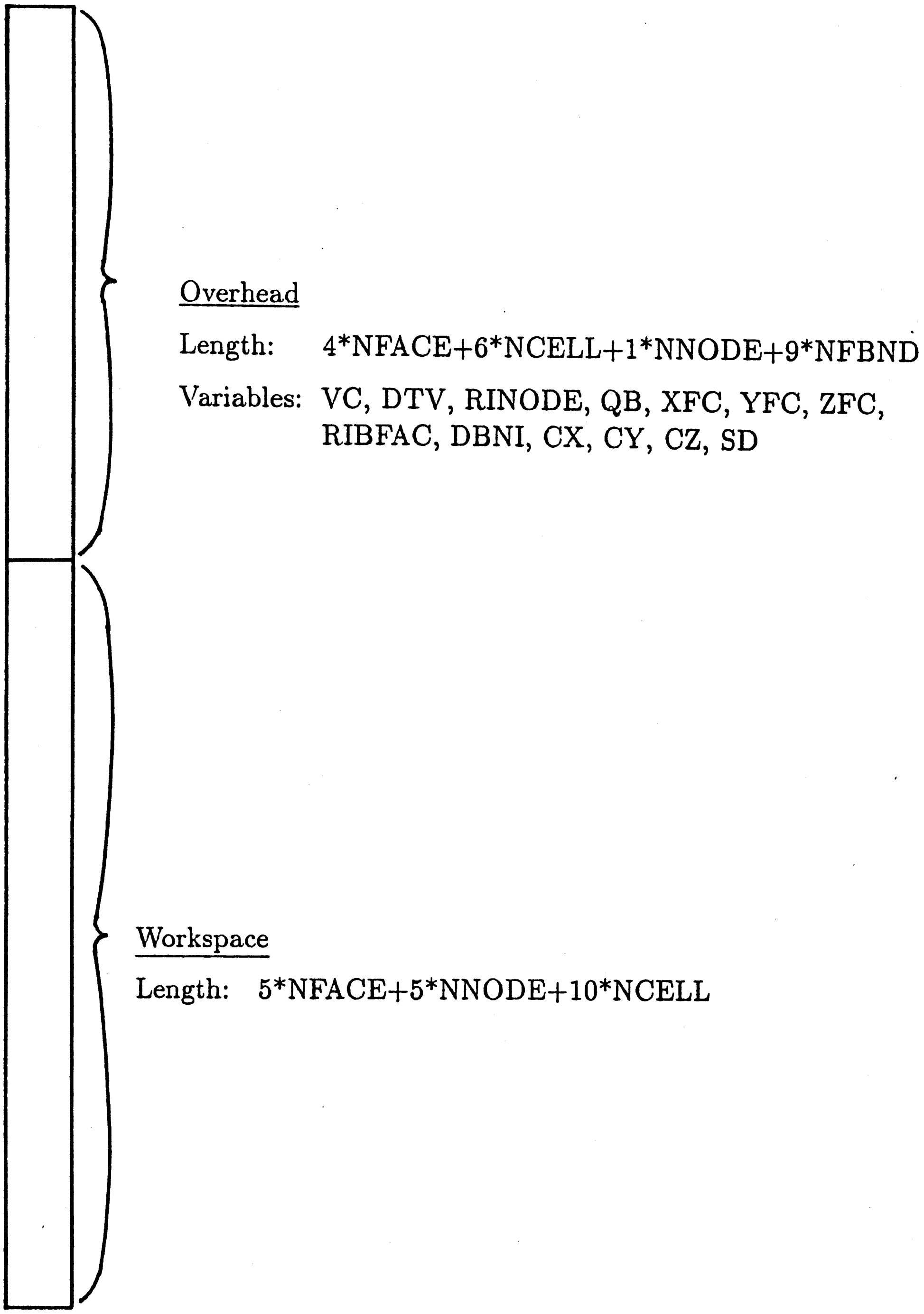
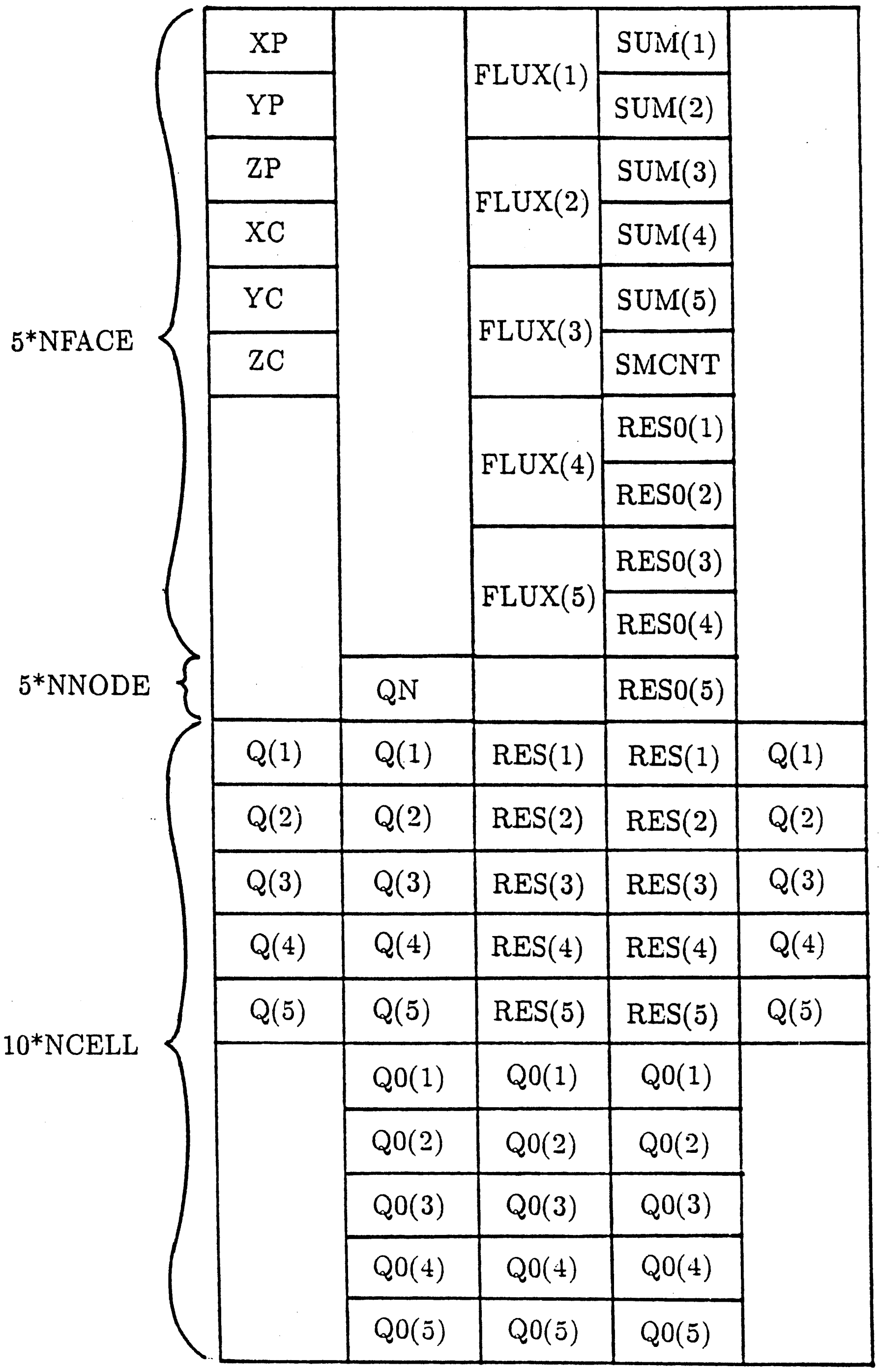


Figure 1.- Tetrahedral grid in unit cube for testing gradient computation. 2020 nodes, 9906 cells.


Overhead
Length: 4*NFACE+6*NCELL+1*NNODE+9*NFBND
Variables: VC, DTV, RINODE, QB, XFC, YFC, ZFC, RIBFAC, DBNI, CX, CY, CZ, SD
Workspace
Length: 5*NFACE+5*NNODE+10*NCELL

(a) Entire array "W".

Figure 2.- Schematic of memory utilization for real variables.


5*NFACE
5*NNODE
10*NCELL
XP
YP
ZP
XC
YC
ZC
QN
FLUX(1)
FLUX(2)
FLUX(3)
FLUX(4)
FLUX(5)
SUM(1)
SUM(2)
SUM(3)
SUM(4)
SUM(5)
SMCNT
RES0(1)
RES0(2)
RES0(3)
RES0(4)
RES0(5)
Q(1)
Q(2)
Q(3)
Q(4)
Q(5)
RES(1)
RES(2)
RES(3)
RES(4)
RES(5)
Q0(1)
Q0(2)
Q0(3)
Q0(4)
Q0(5)

(b) Workspace.

Figure 2.- Concluded.

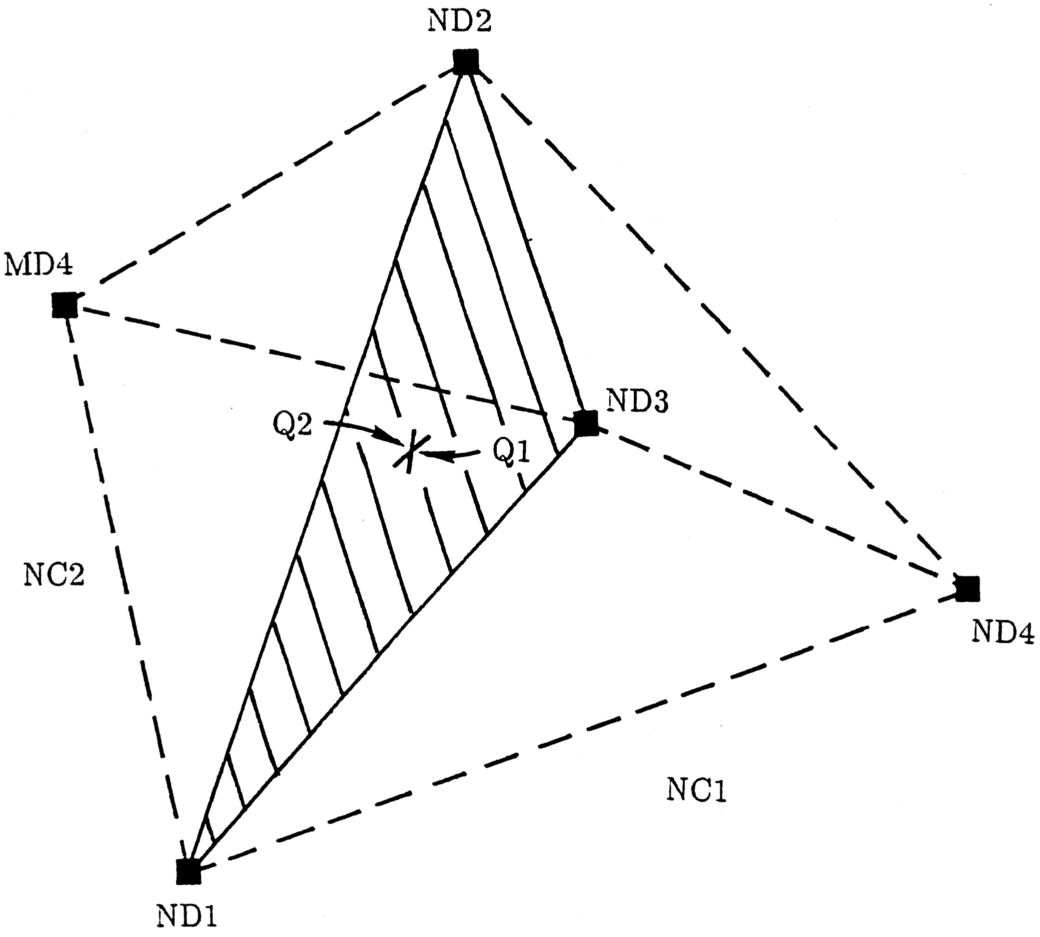
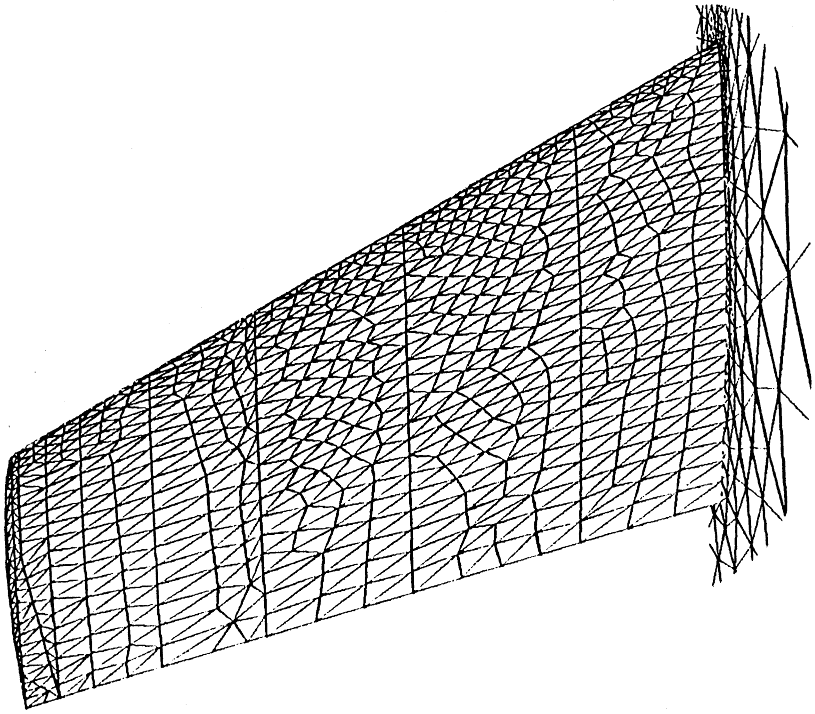
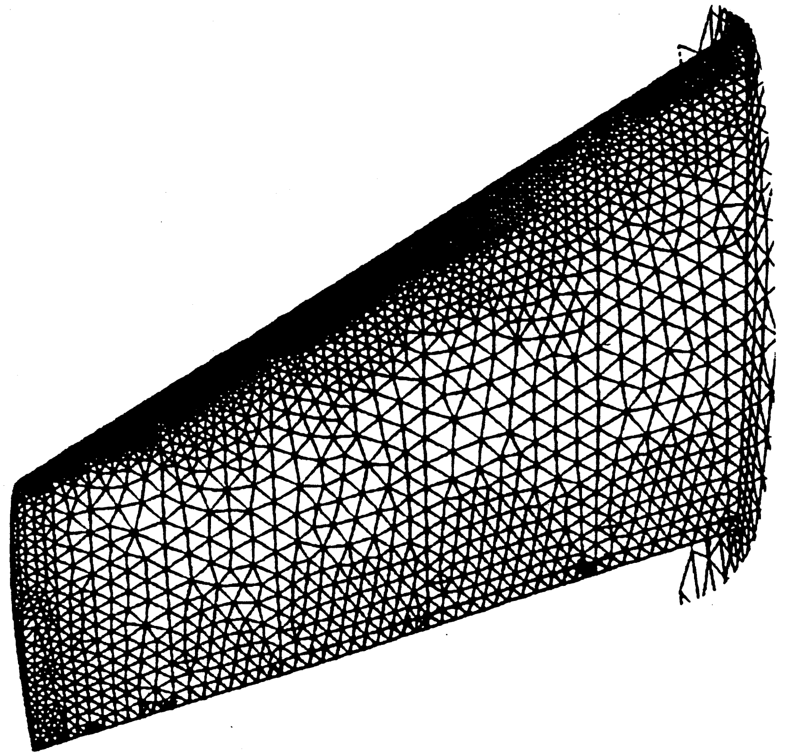

ND2
MD4
ND3
Q2
Q1
NC2
ND4
NC1
ND1

Figure 3.- Two cells sharing common face.



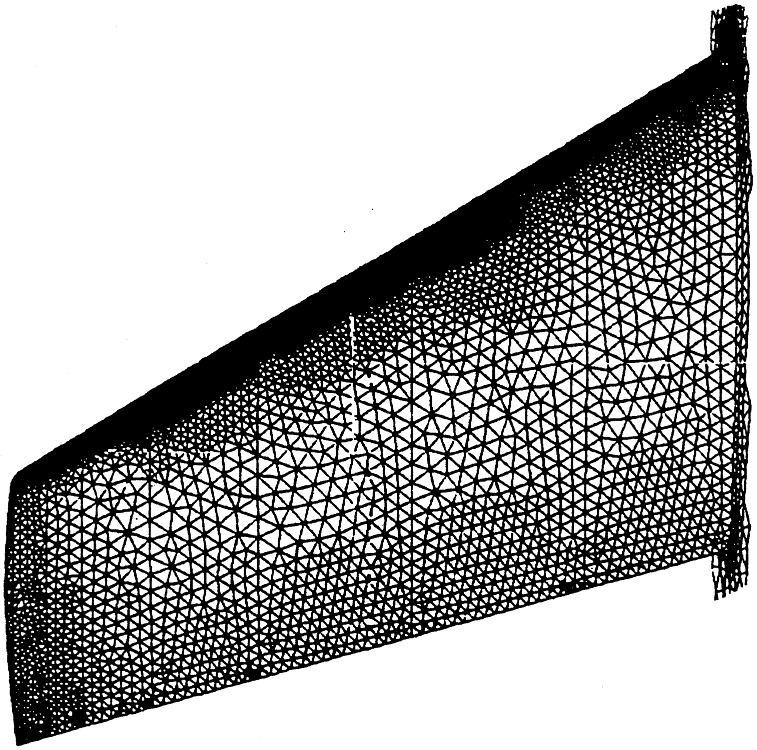
(a) Mesh 1.

Figure 4.- Upper surface mesh for ONERA M6 wing.



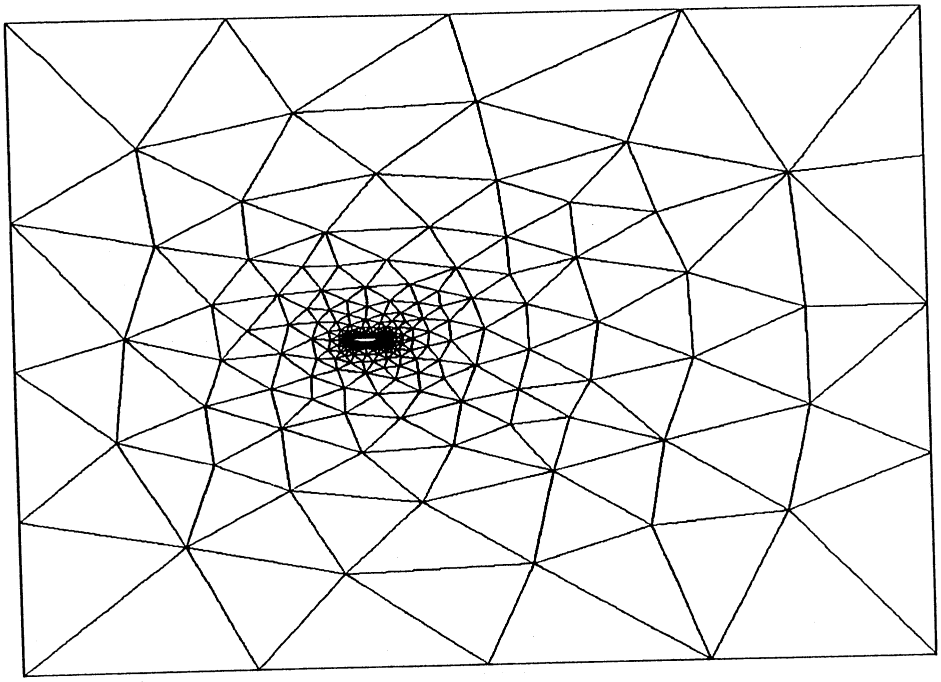
(b) Mesh 2.

Figure 4.- Continued.

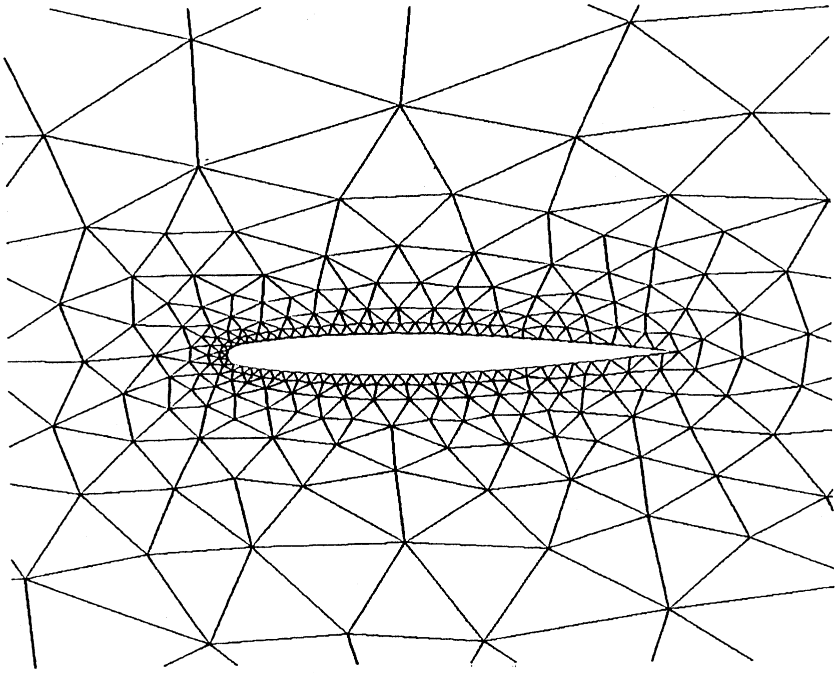


(c) Mesh 3.

Figure 4.- Concluded.

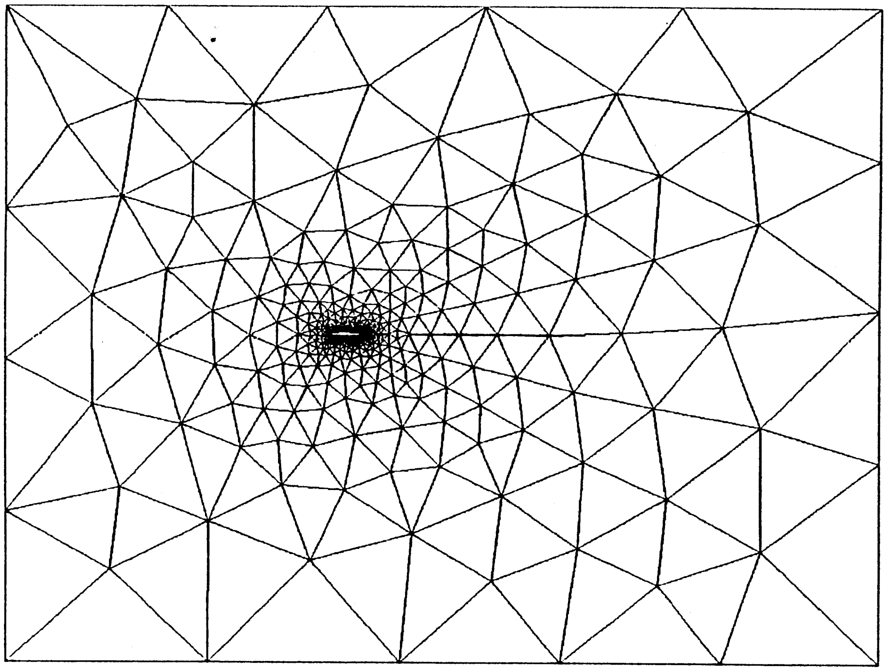


(a) Mesh 1 (Farfield).

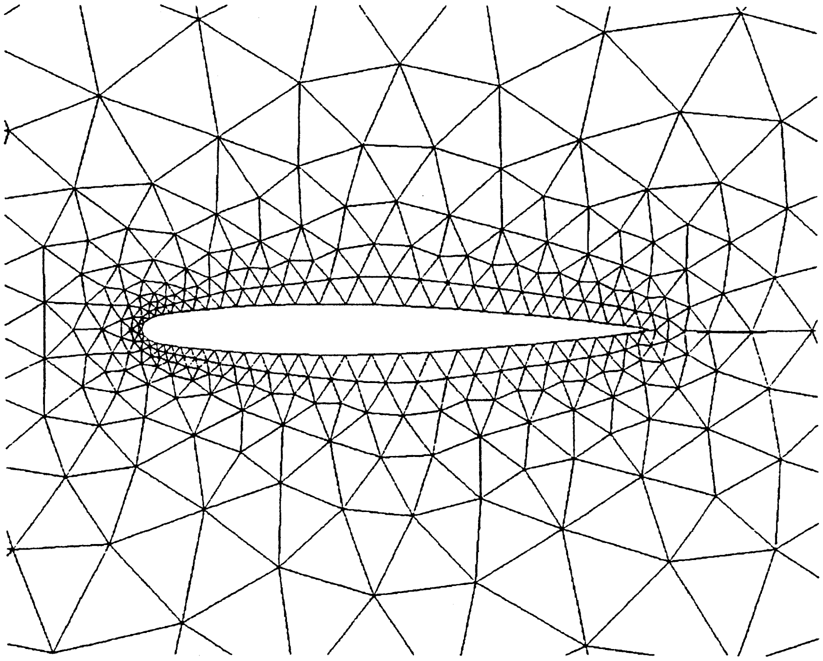


(b) Mesh 1 (Nearfield).

Figure 5.- Mesh at symmetry plane for ONERA M6 wing.

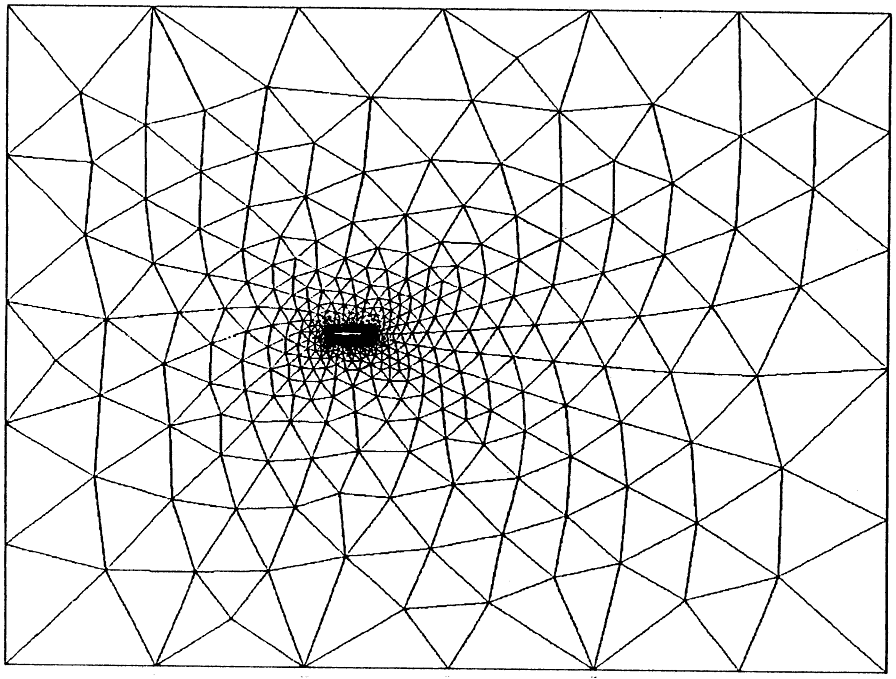


(c) Mesh 2 (Farfield).

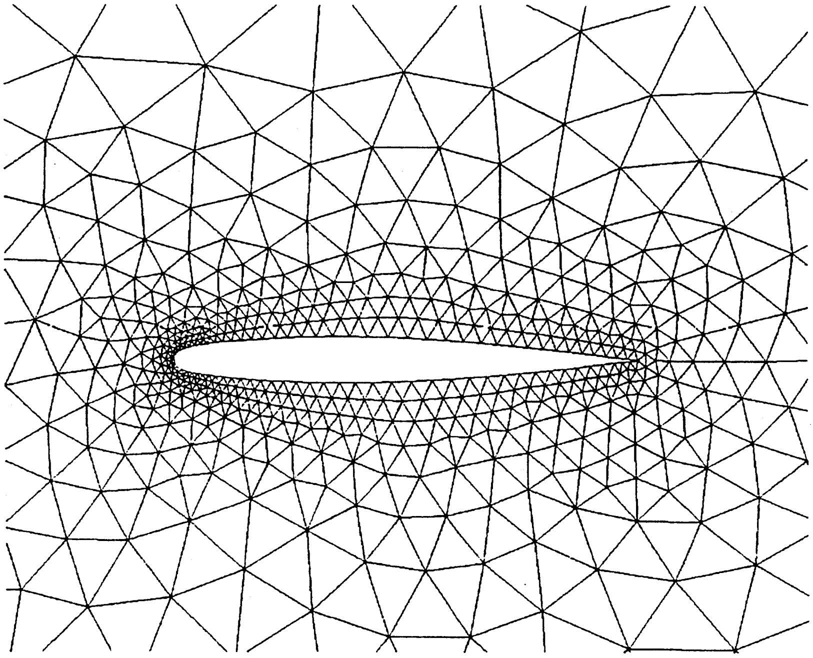


(d) Mesh 2 (Nearfield).

Figure 5.- Continued.

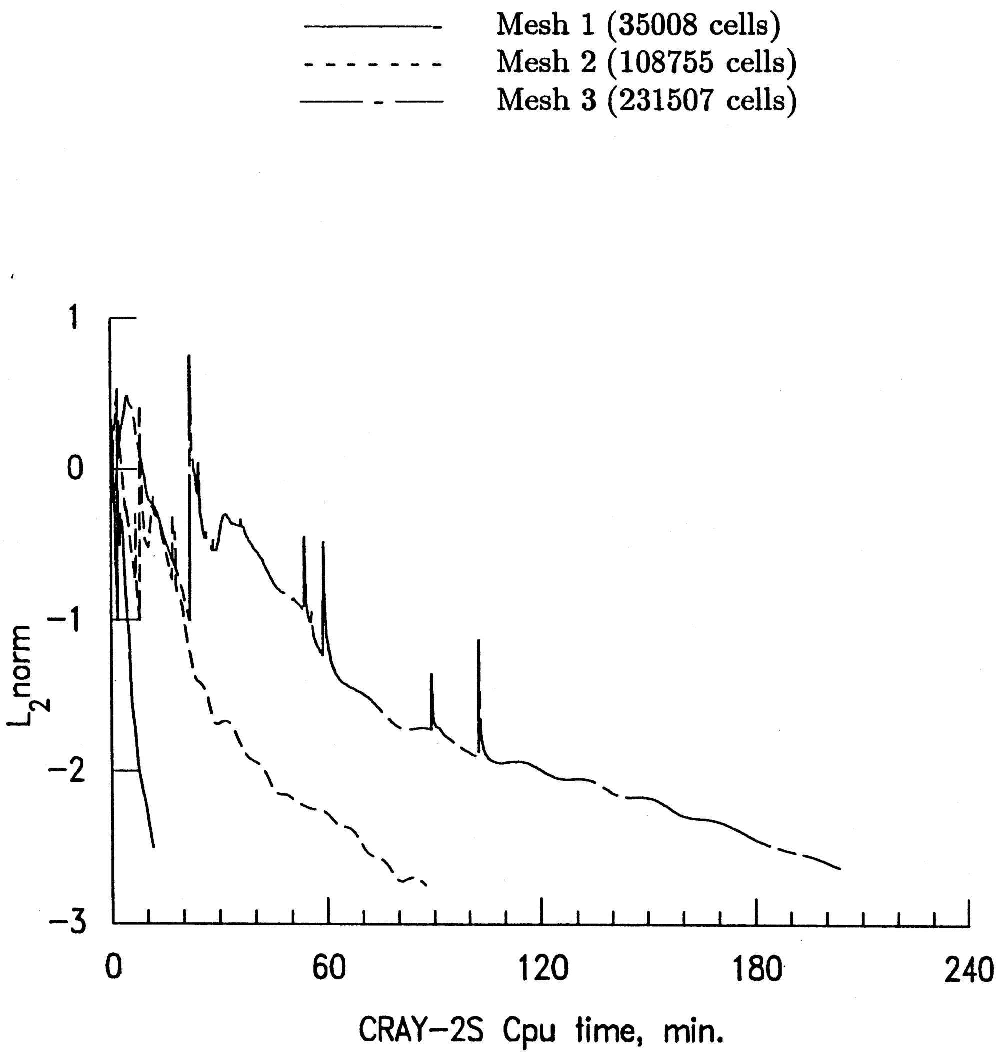


(e) Mesh 3 (Farfield).



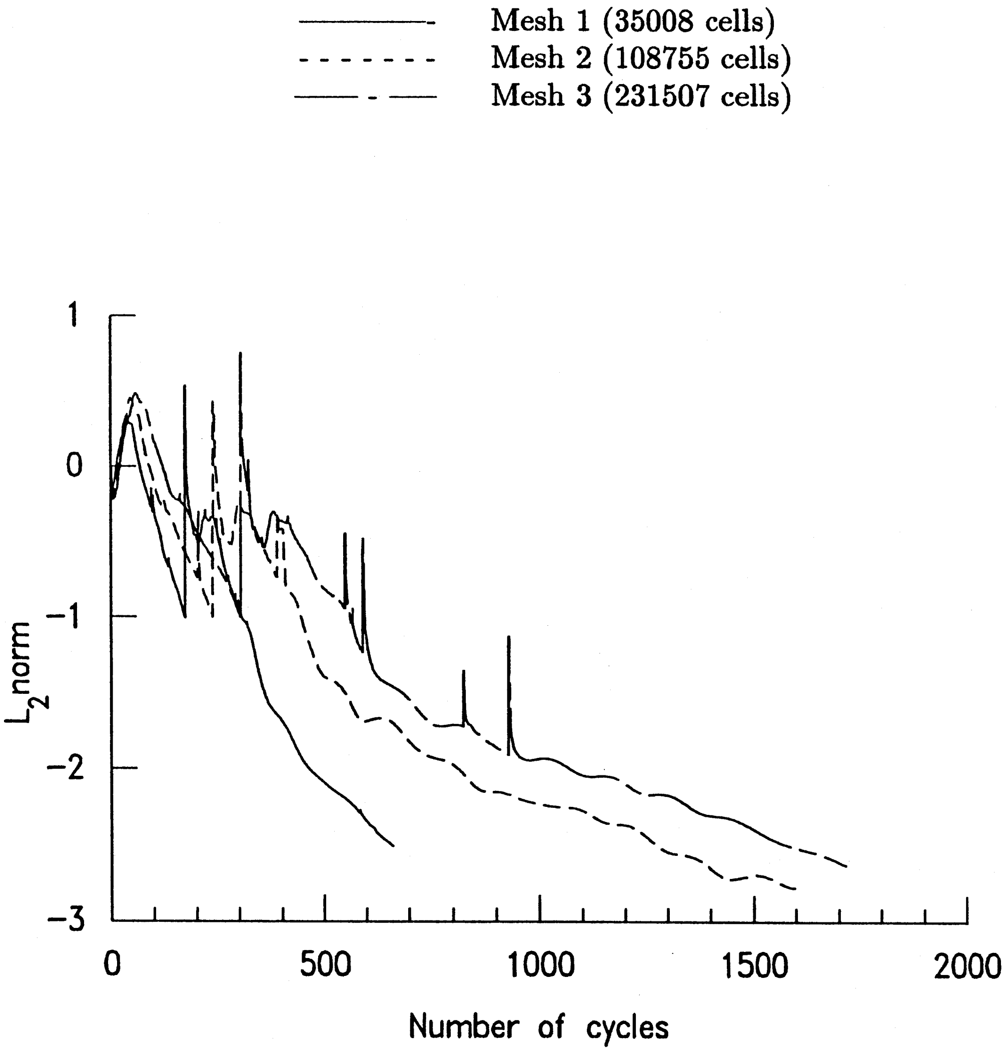
(f) Mesh 3 (Nearfield).

Figure 5.- Concluded.



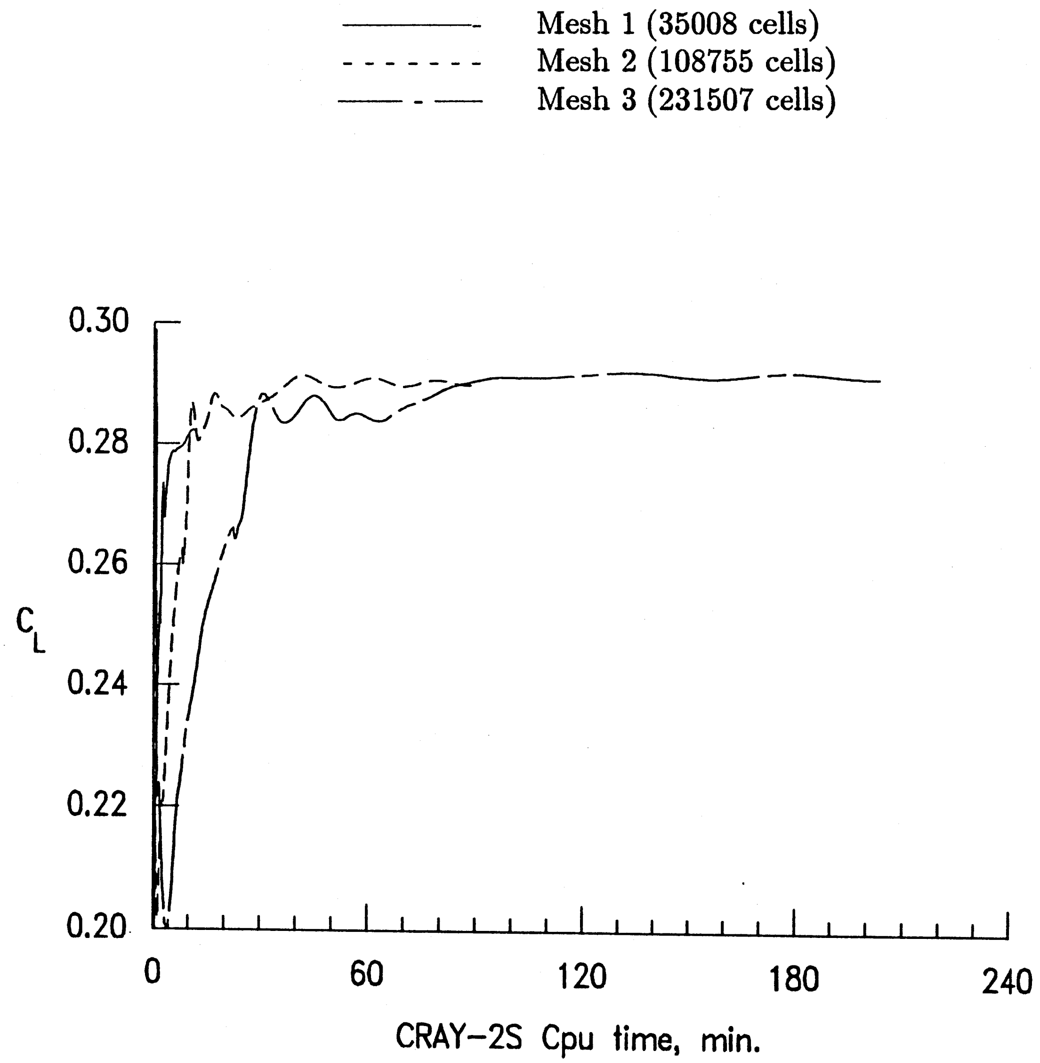
(a) L_2-norm vs. CPU time

Figure 6.- Effect of mesh size on convergence history of ONERA M6 wing. $M_\infty = 0.84, \alpha = 3.06°$.



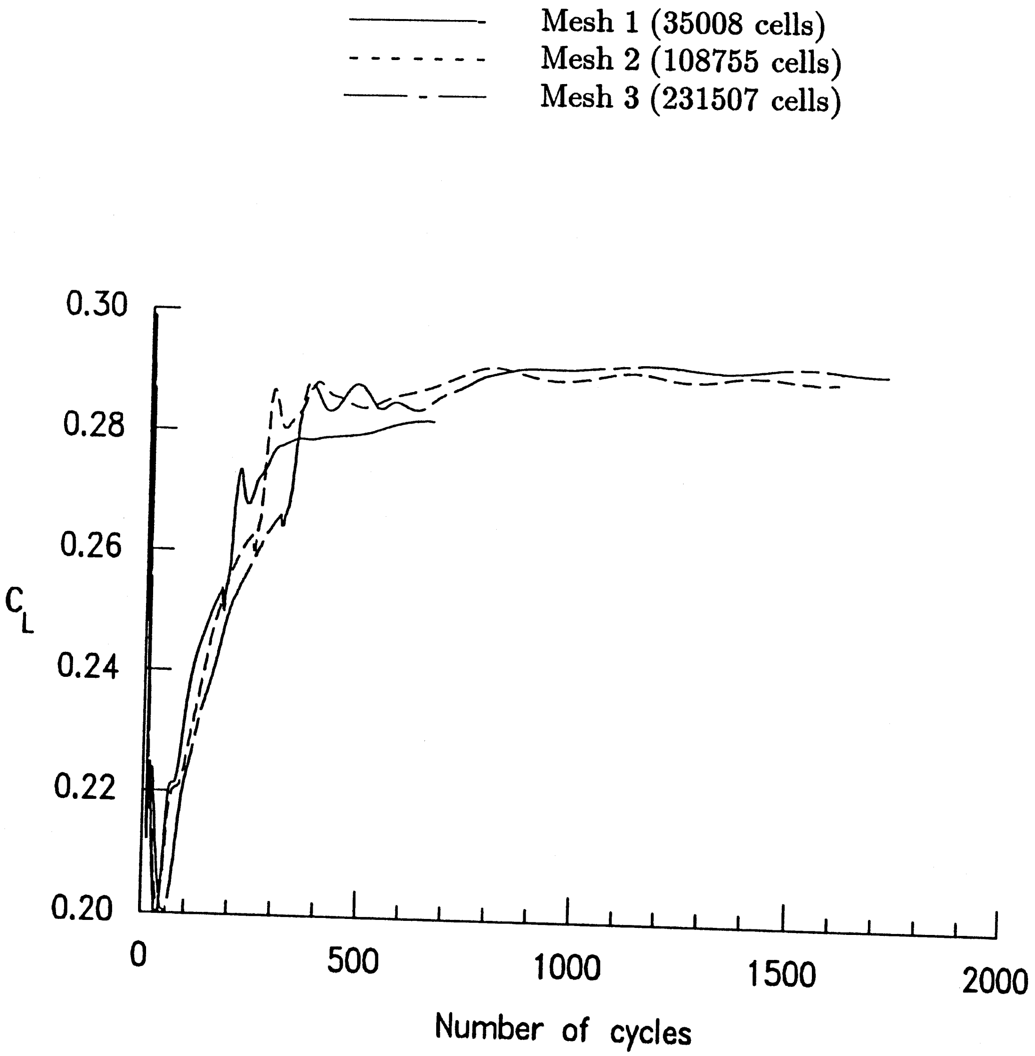
(b) L_2-norm vs. Iteration

Figure 6.- Continued.



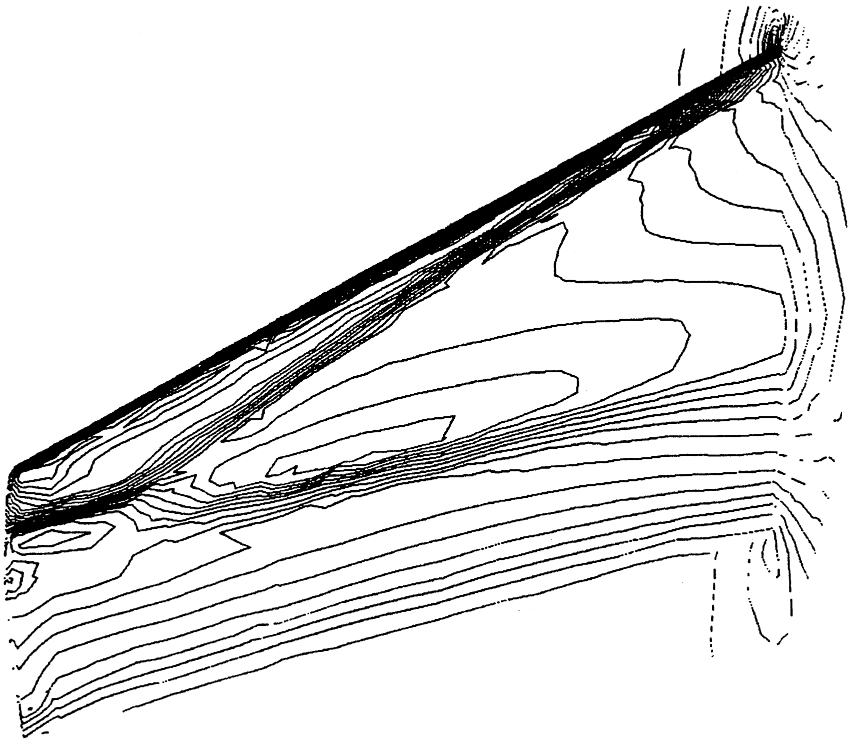
(c) Lift coefficient vs. CPU time

Figure 6.- Continued.

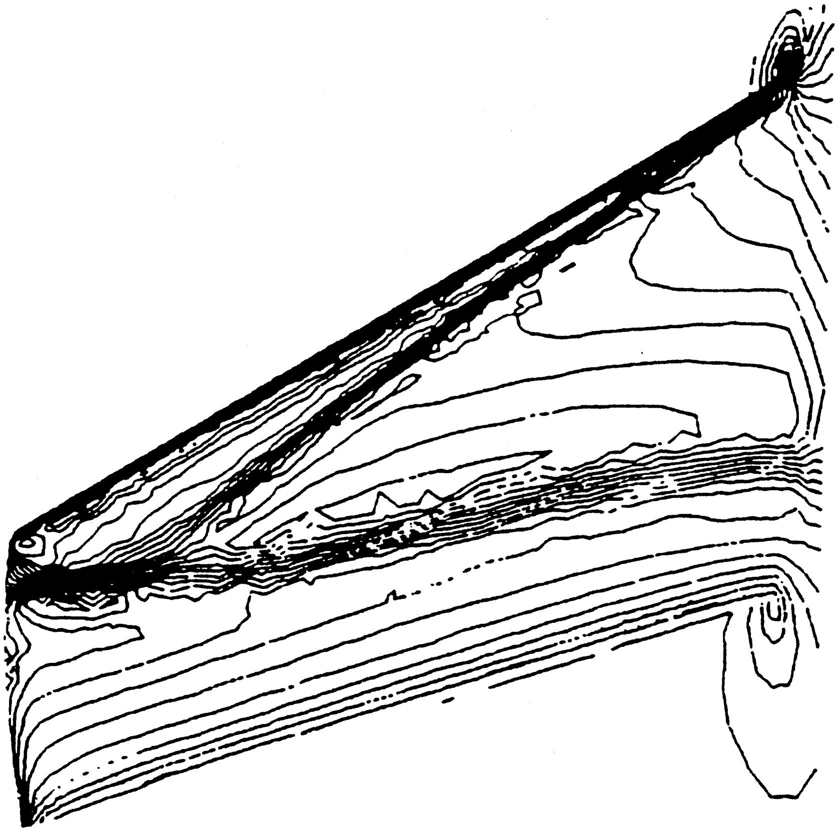


(d) Lift coefficient vs. Iteration

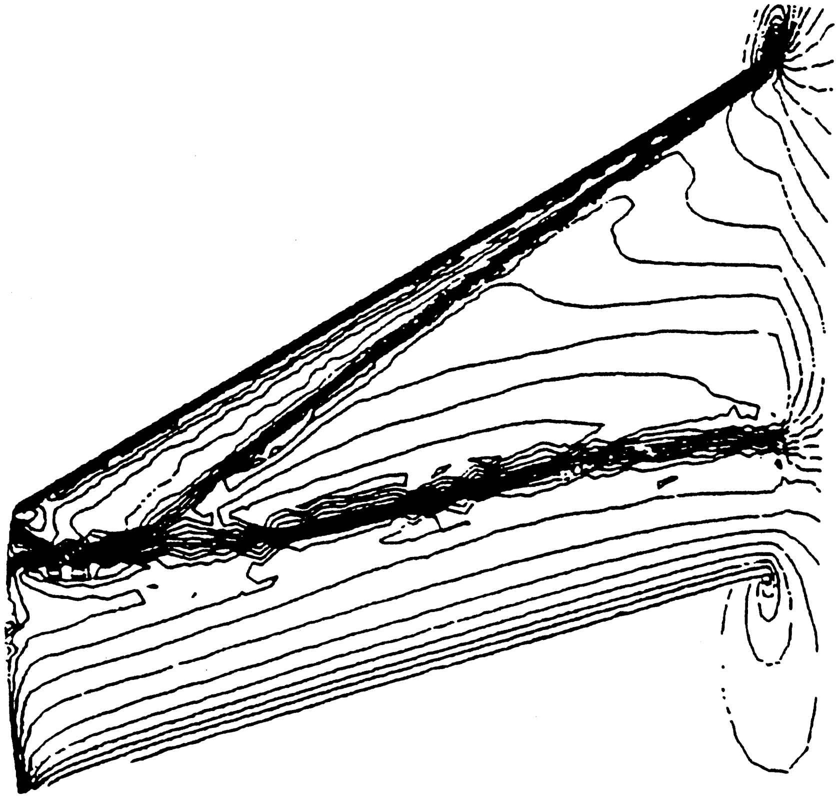
Figure 6.- Concluded.



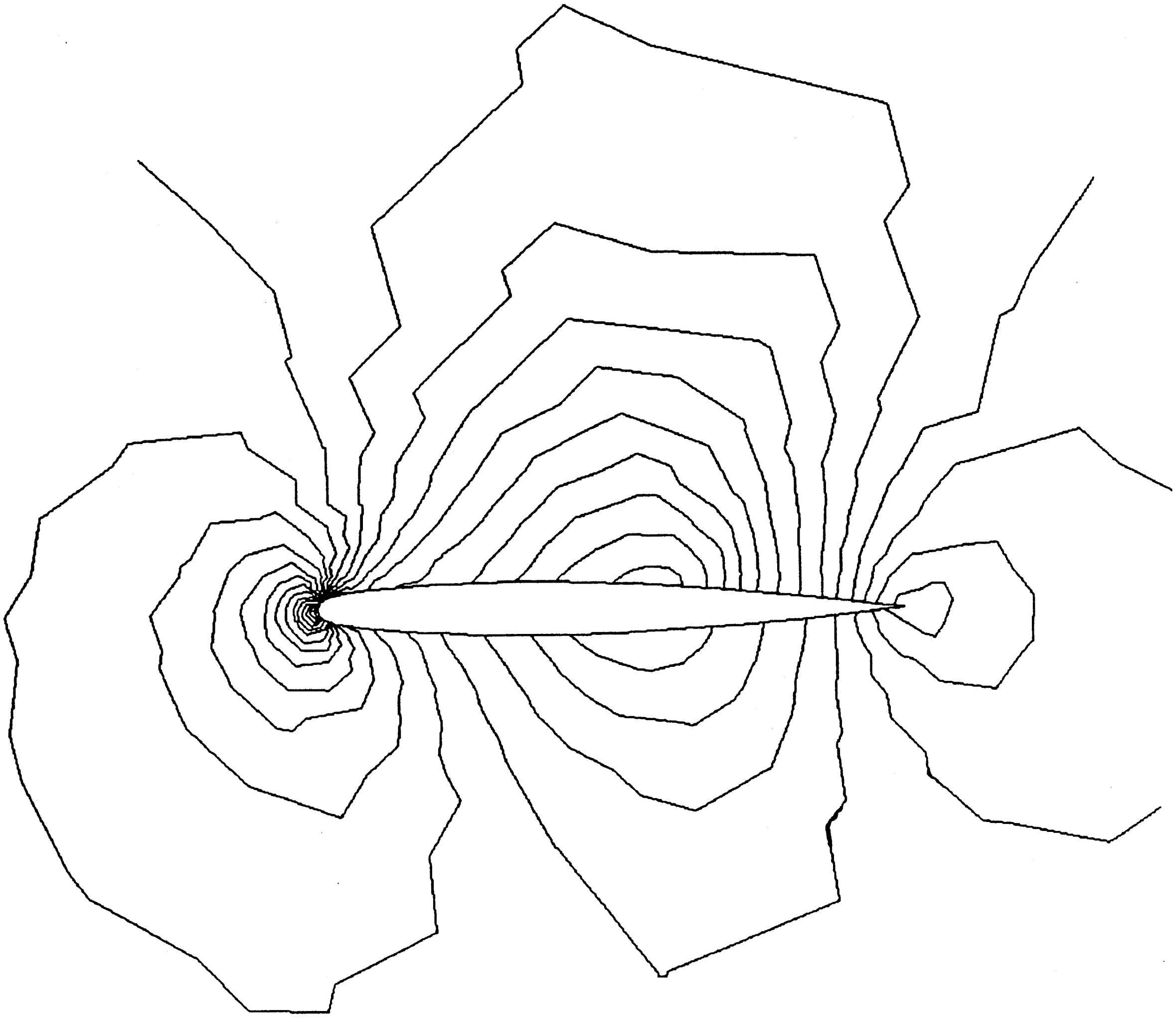
(a) Mesh 1.
Figure 7.- Upper surface pressure contours for ONERA M6 wing.
$M_\infty = 0.84,\ \alpha = 3.06°,\ \Delta(p/p_\infty) = 0.02.$



(b) Mesh 2.
Figure 7.- Continued.



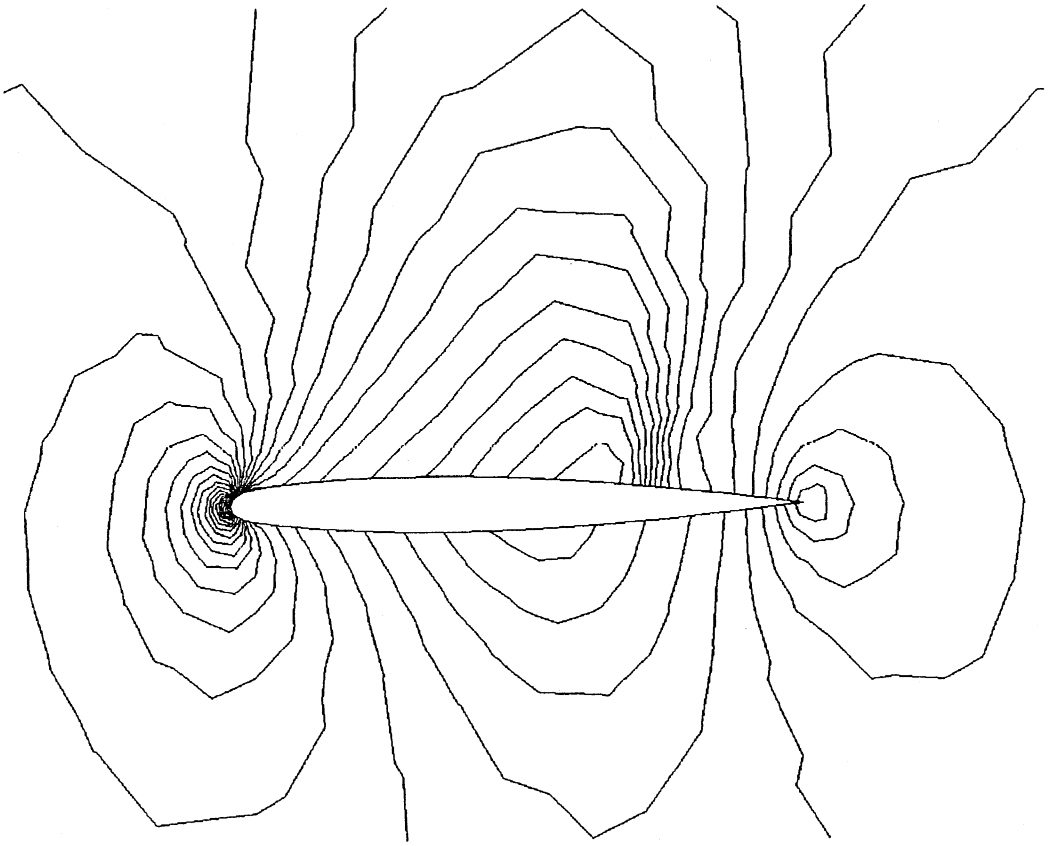
(c) Mesh 3.
Figure 7.- Concluded.



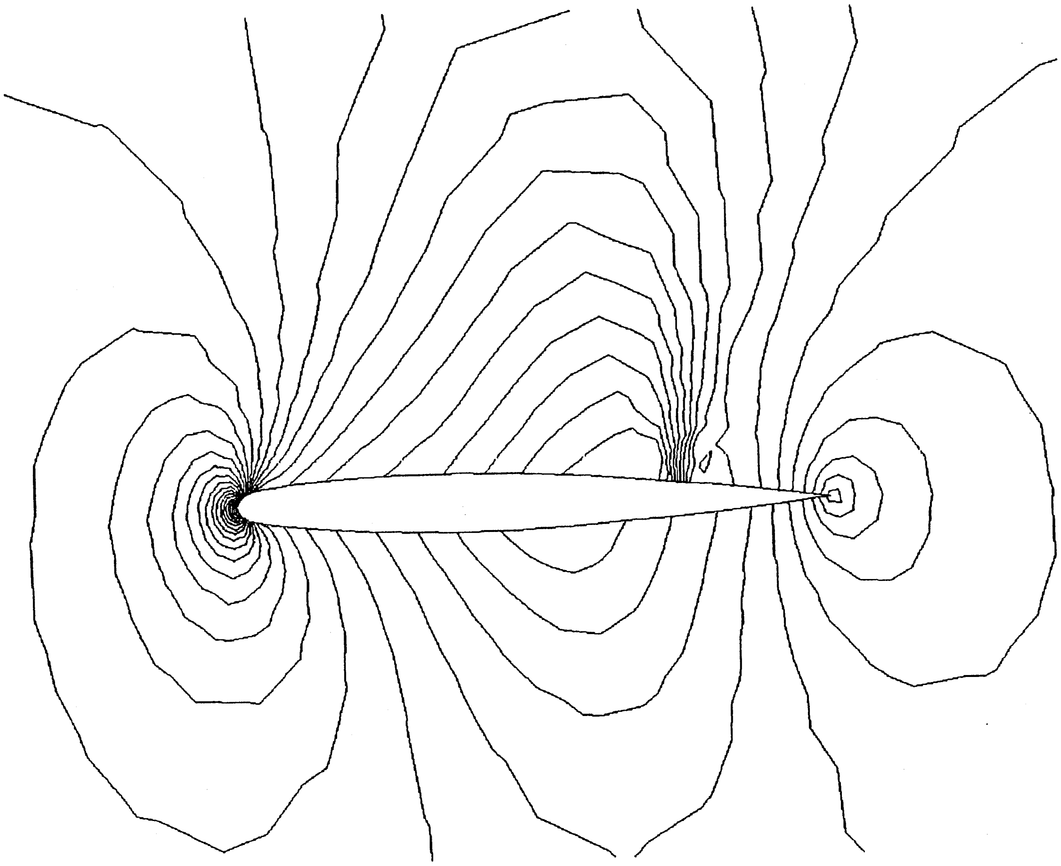
(a) Mesh 1.

Figure 8.- Pressure contours at symmetry plane for ONERA M6 wing.

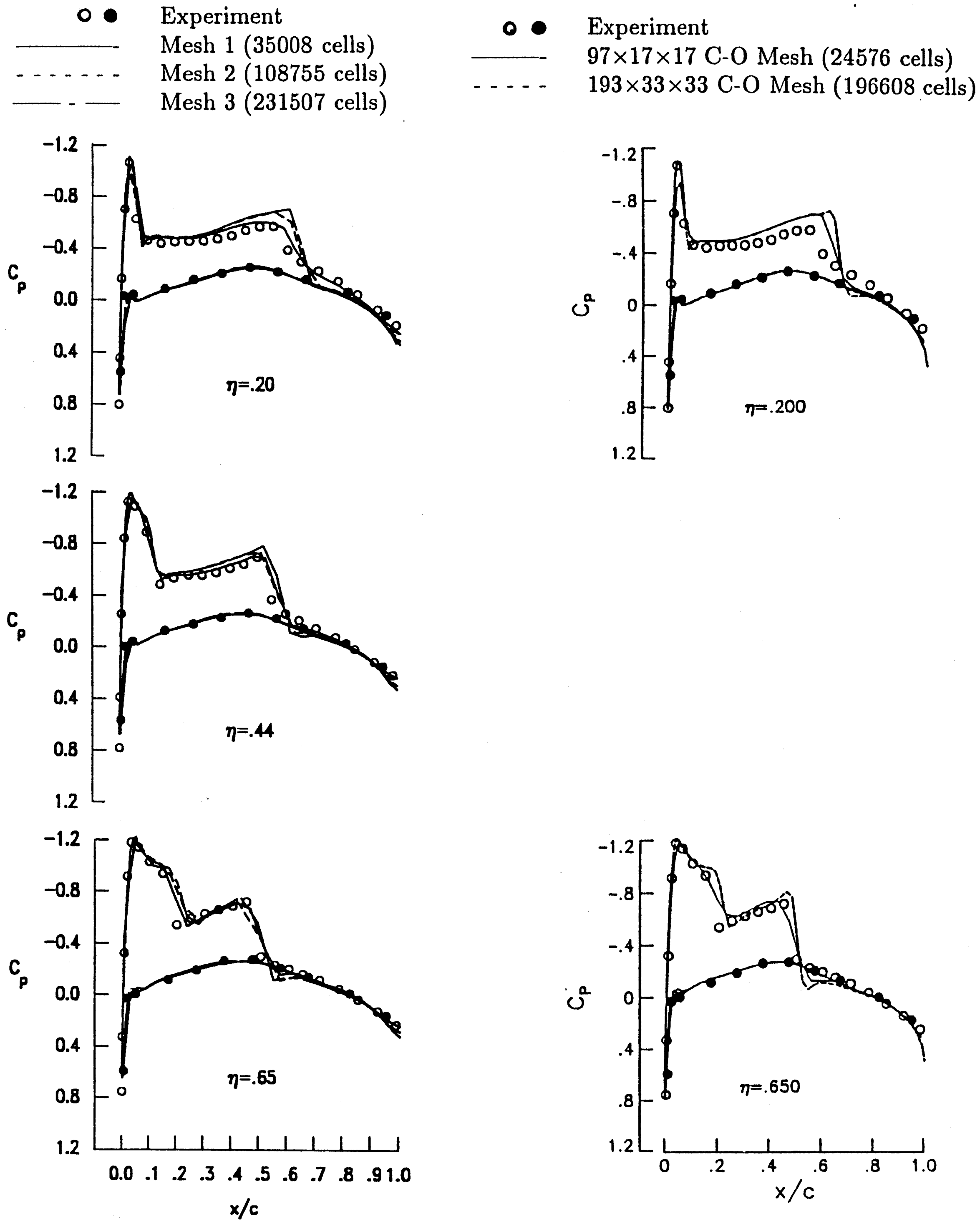
$M_\infty = 0.84$, $\alpha = 3.06°$, $\Delta(p/p_\infty) = 0.02$.



(b) Mesh 2.
Figure 8.- Continued.



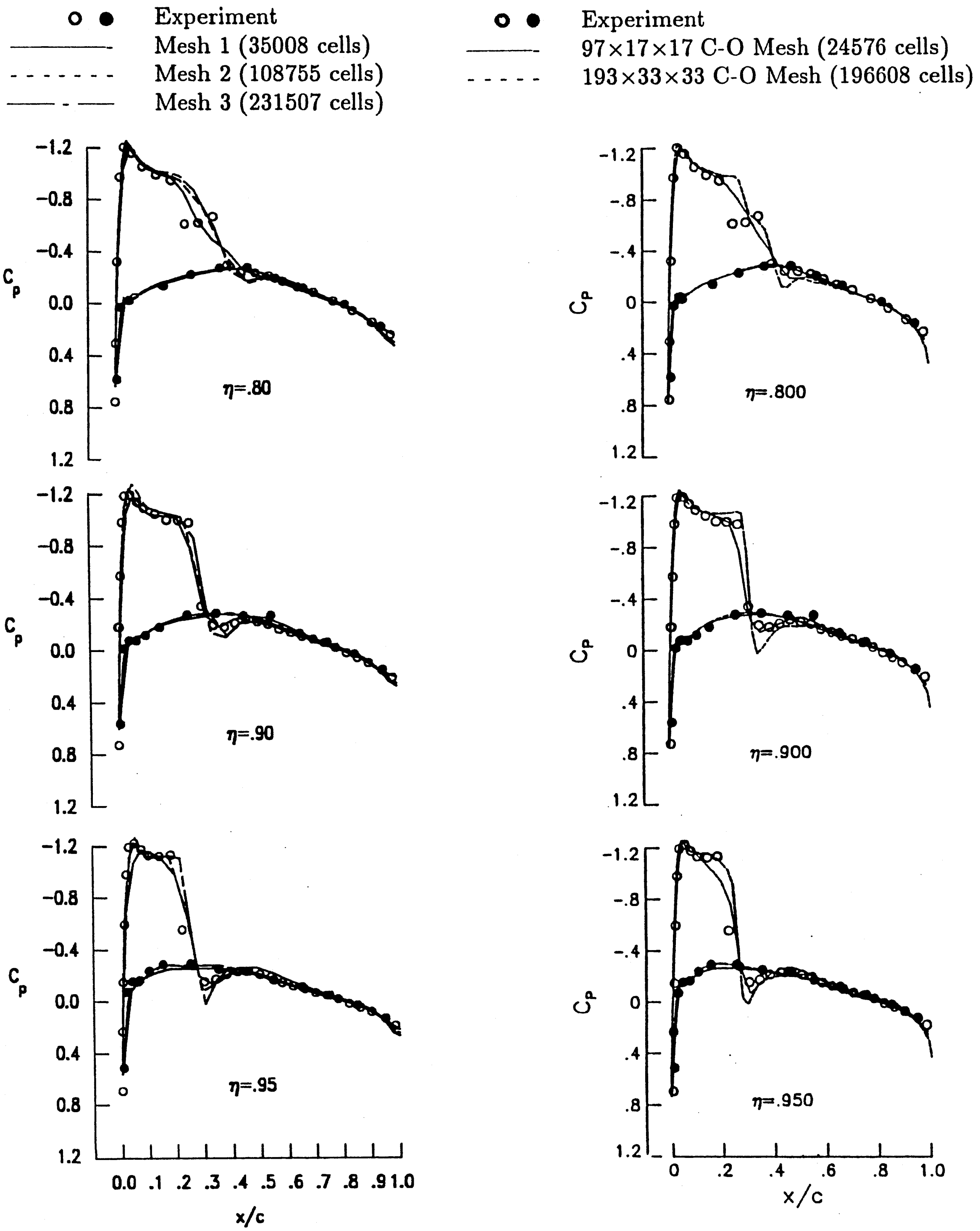
(c) Mesh 3.
Figure 8.- Concluded.



(a) Unstructured.

(b) Structured (Ref. [80]).

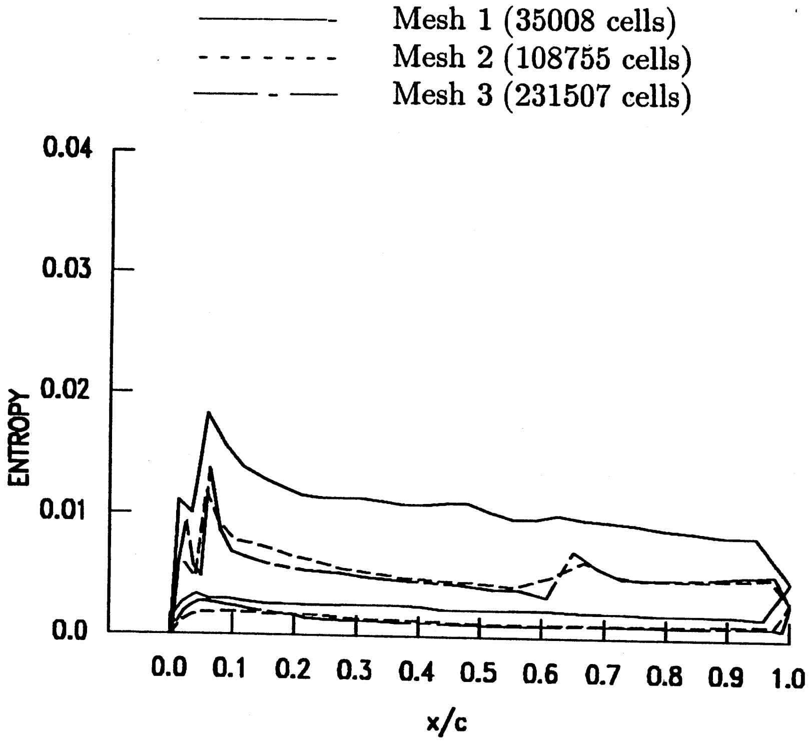
Figure 9.- Comparison of experimental data with unstructured and structured inviscid solutions. ONERA M6 Wing. $M_\infty = 0.84$, $\alpha = 3.06°$.



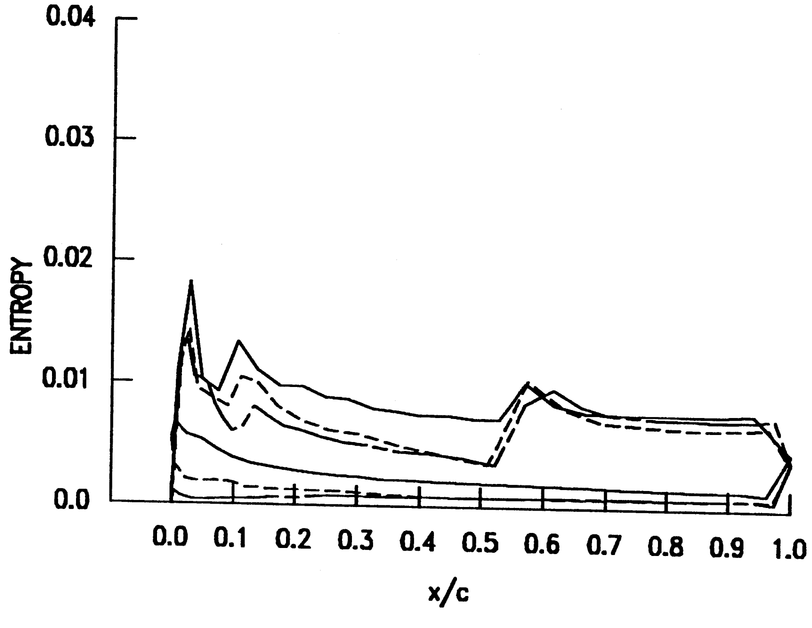
(a) Unstructured.

(b) Structured (Ref. [80]).

Figure 9.- Concluded.

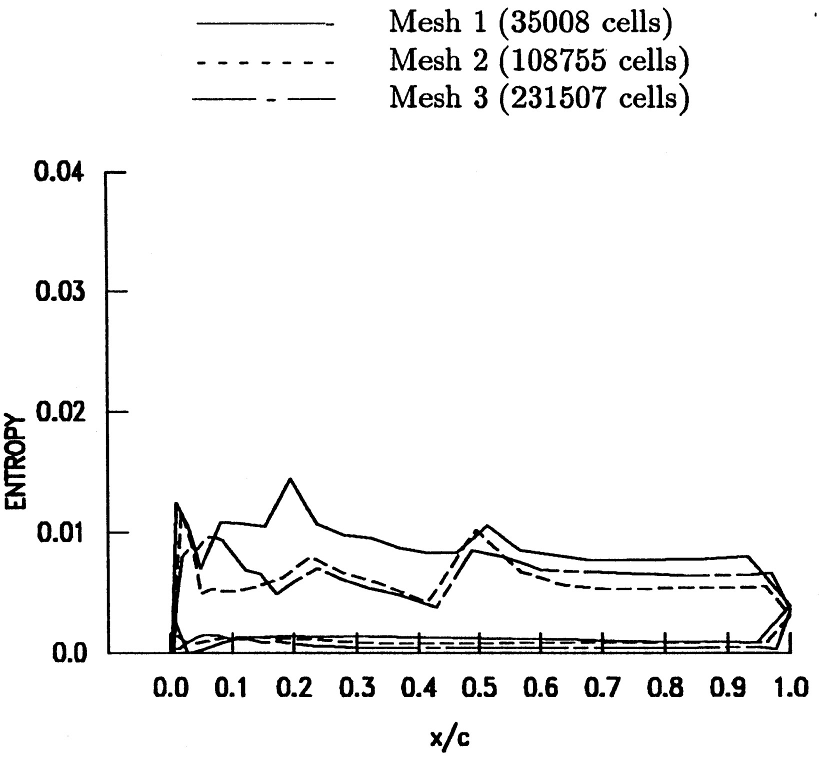


(a) $\eta = 0.20$

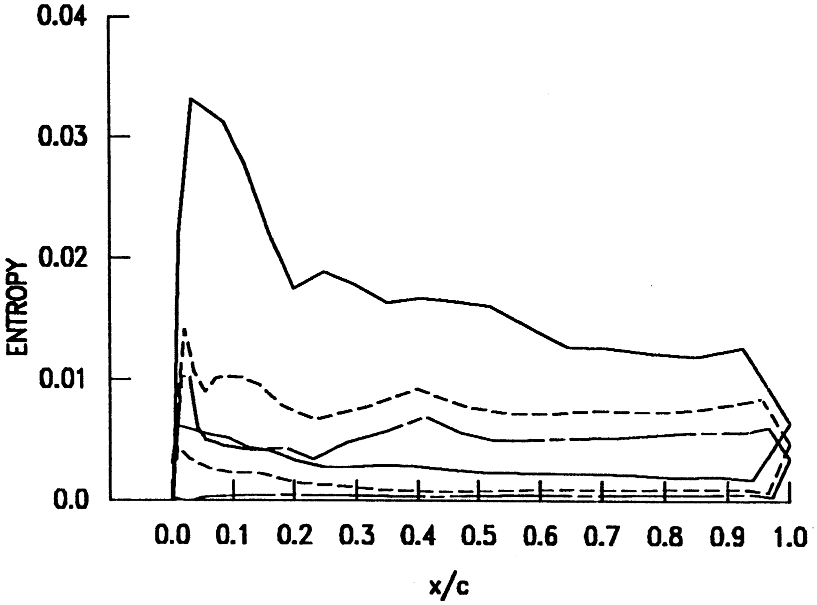


(b) $\eta = 0.44$

Figure 10.- Effect of mesh size on chordwise entropy distribution for ONERA M6 wing. $M_\infty = 0.84, \alpha = 3.06°$.

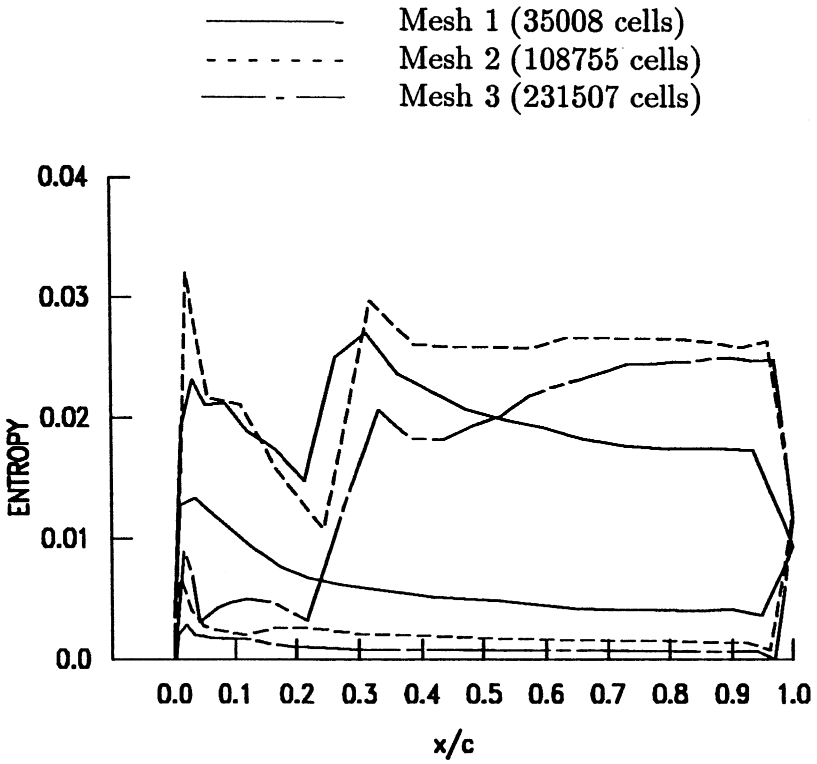


(c) $\eta = 0.65$

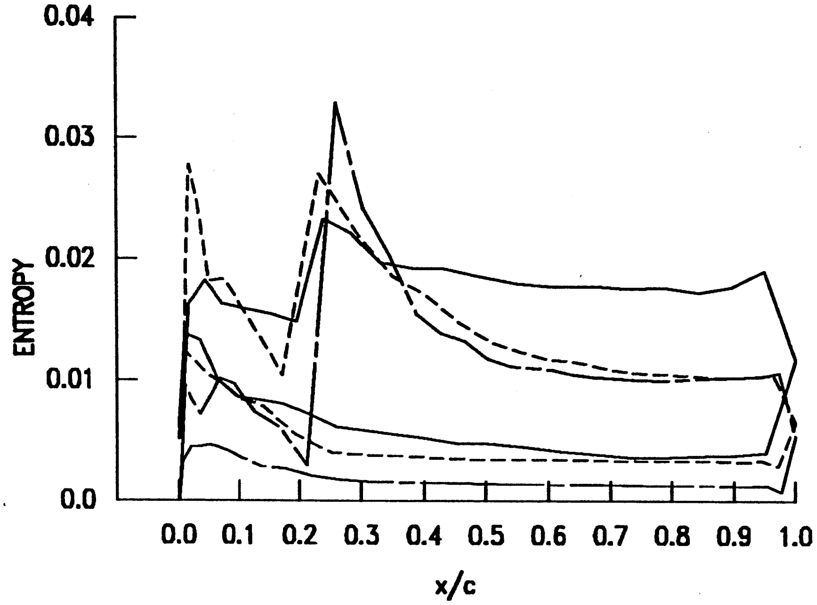


(d) $\eta = 0.80$

Figure 10.- Continued.

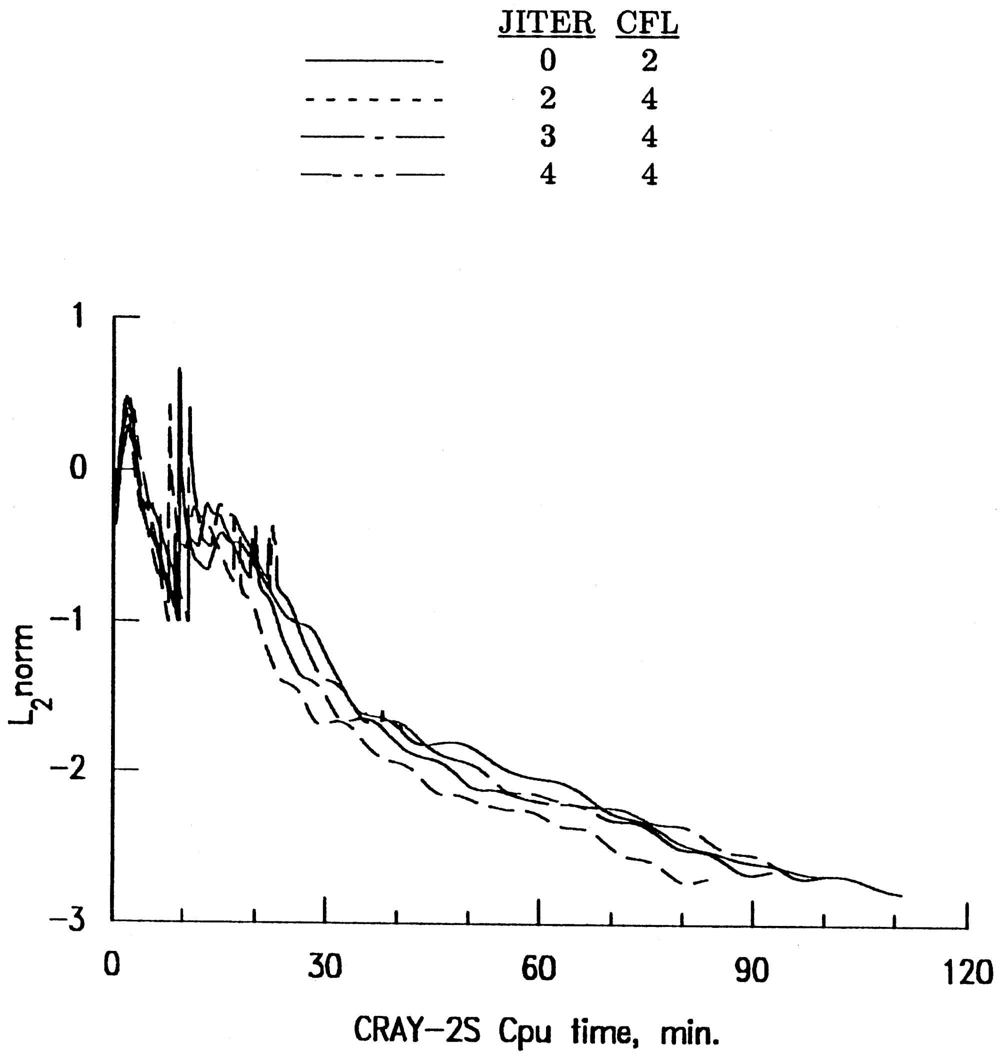


(e) $\eta = 0.90$



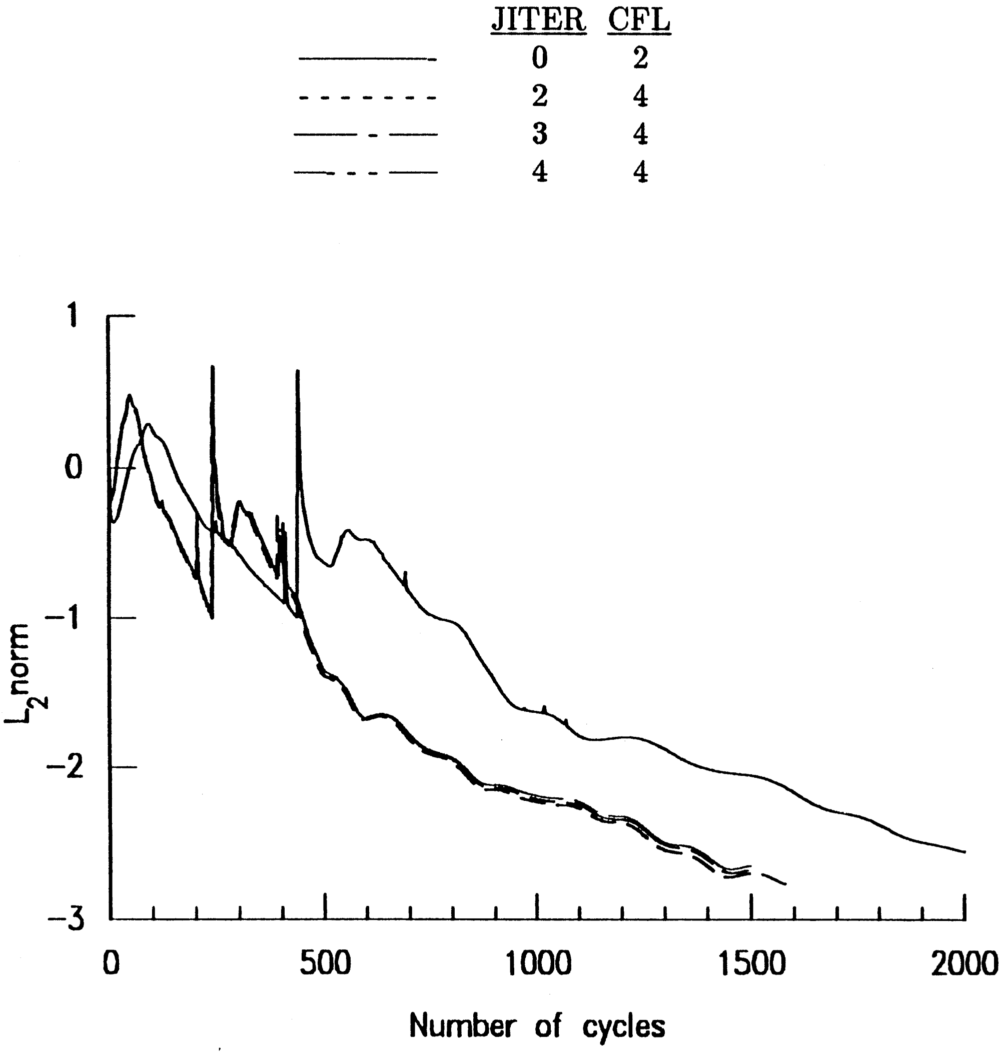
(f) $\eta = 0.95$

Figure 10.- Concluded.



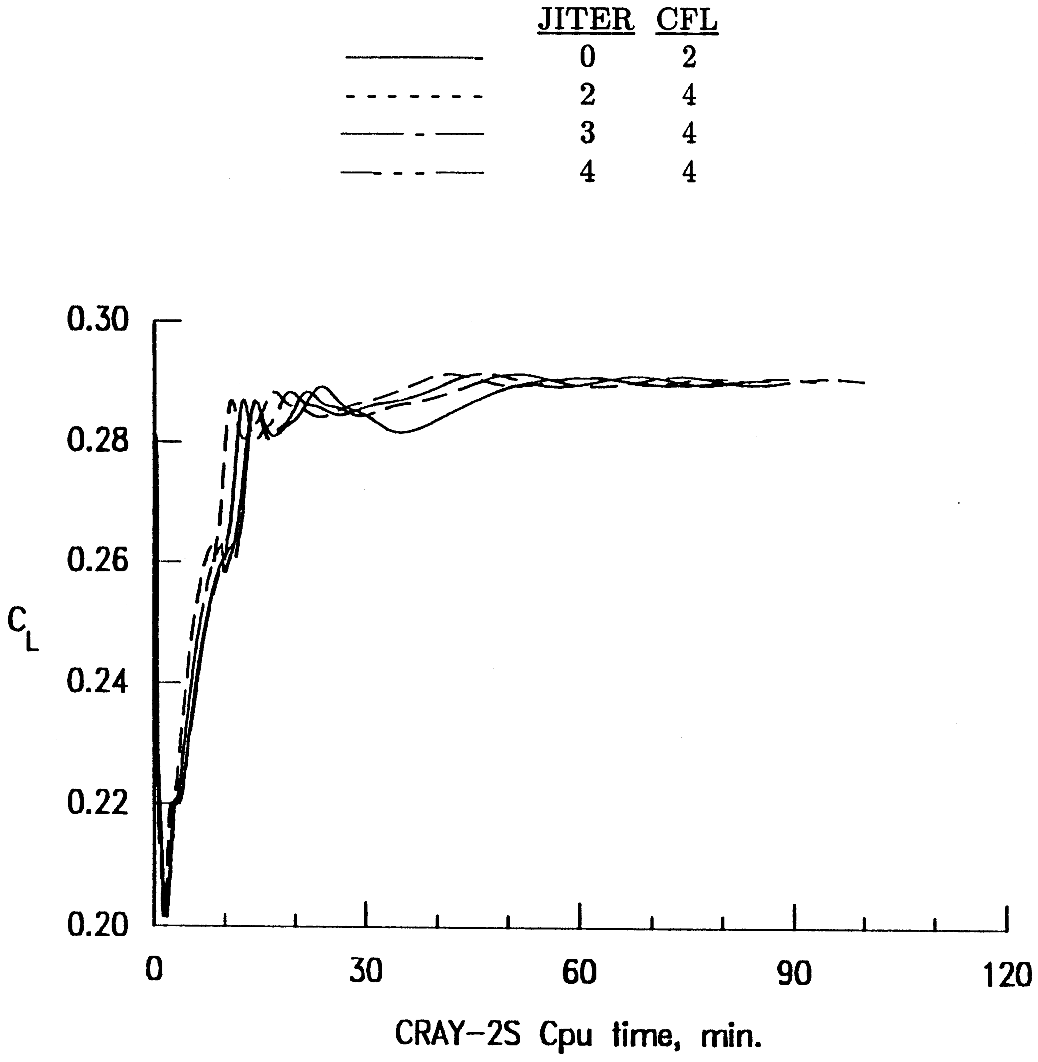
(a) L_2-norm vs. CPU time

Figure 11.- Implicit Residual Smoothing - effect of JITER on convergence history of ONERA M6 wing. $M_\infty = 0.84, \alpha = 3.06°, \varepsilon = 0.5$.



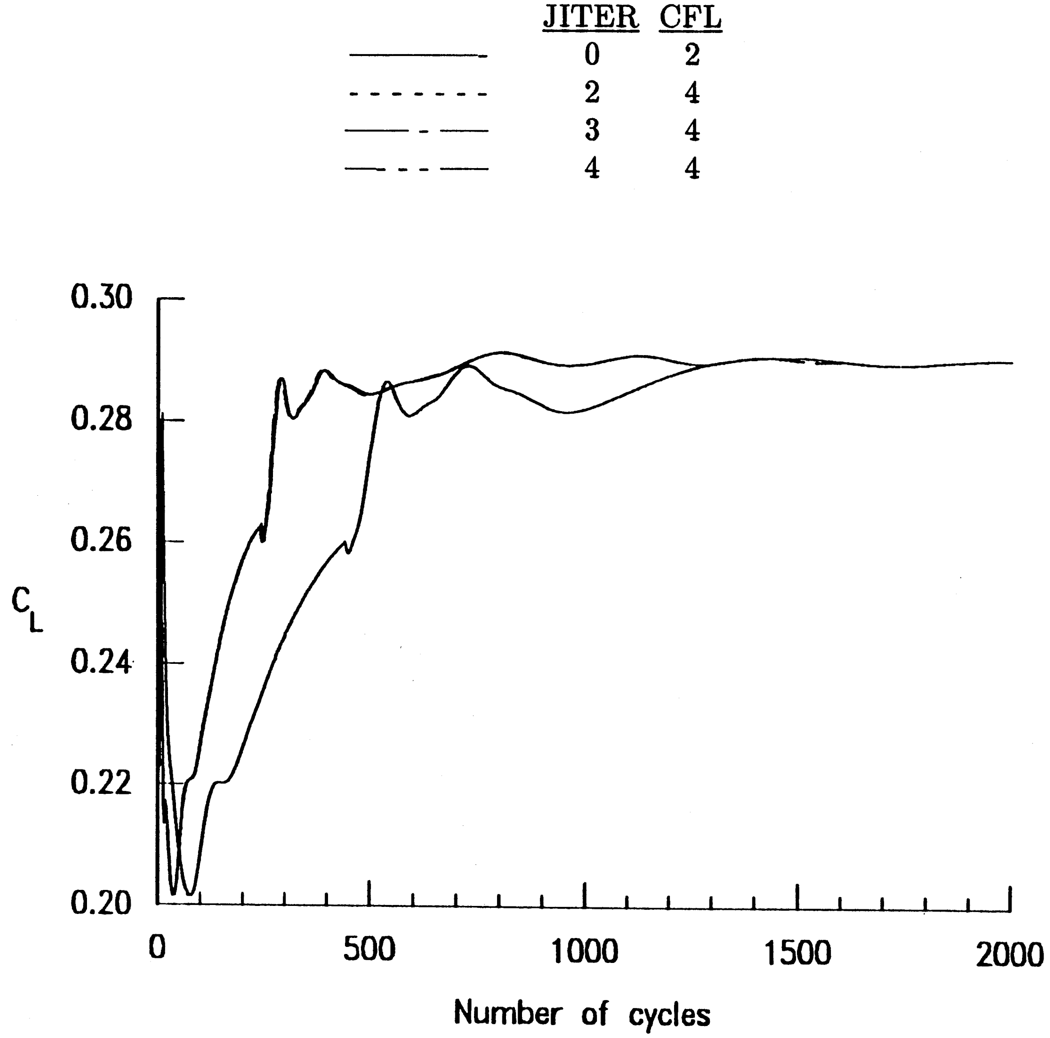
(b) L_2-norm vs. Iteration

Figure 11.- Continued.



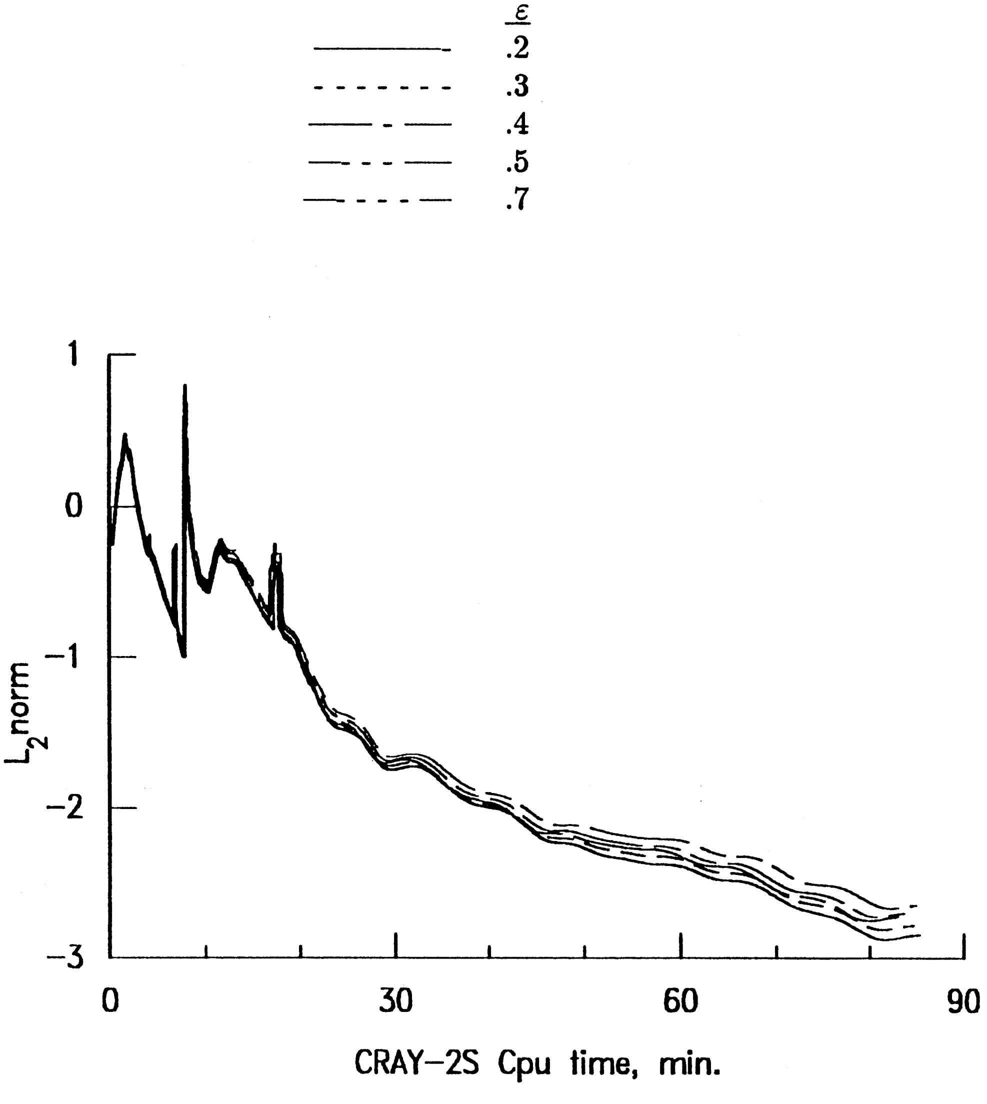
(c) Lift coefficient vs. CPU time

Figure 11.- Continued.



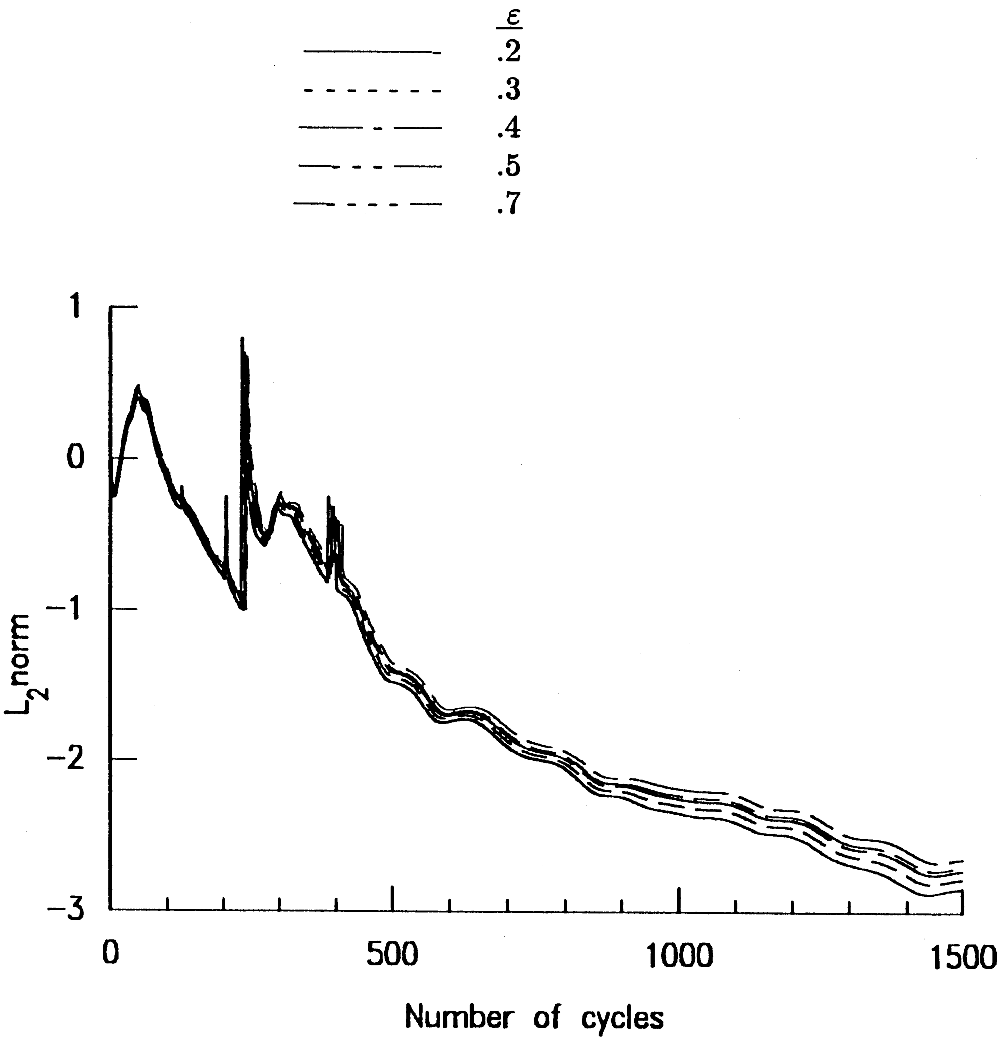
(d) Lift coefficient vs. Iteration

Figure 11.- Concluded.



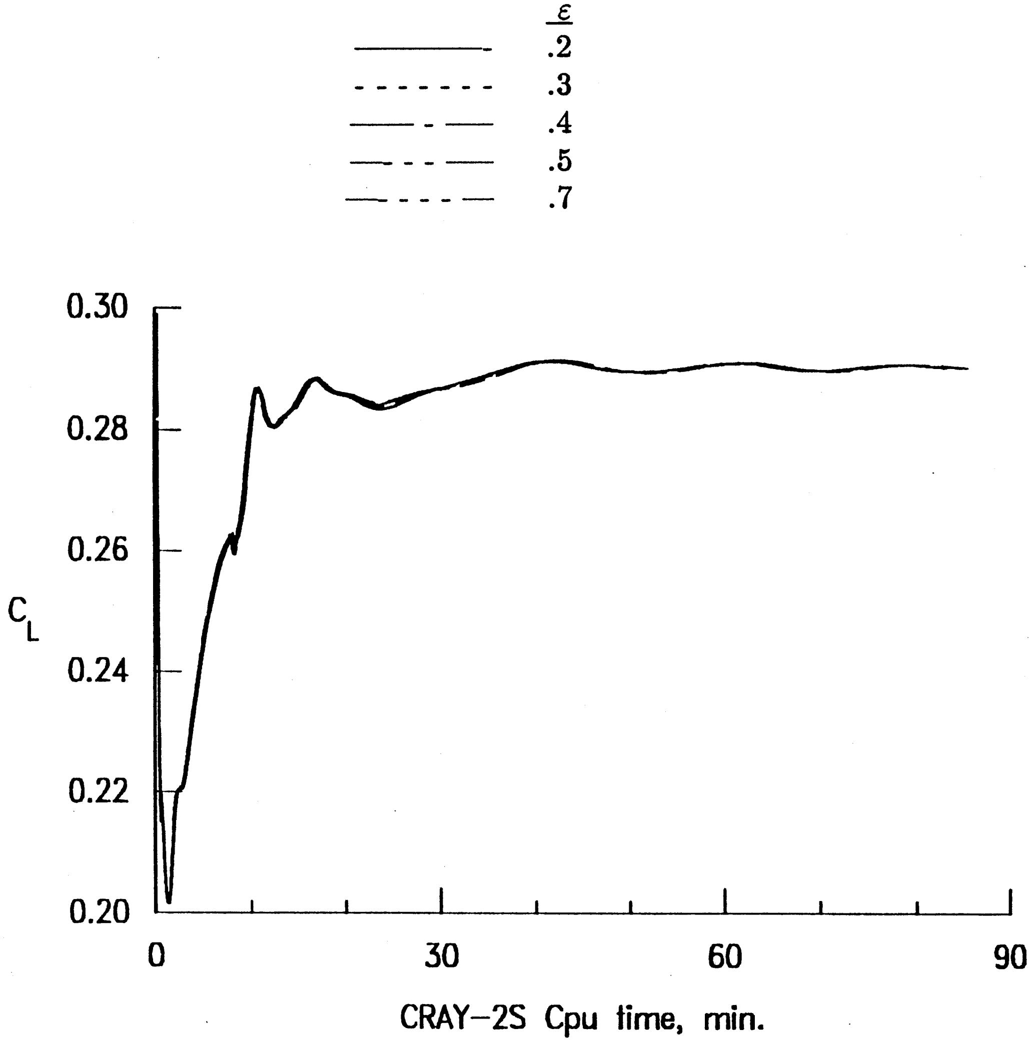
(a) L_2-norm vs. CPU time

Figure 12.- Implicit Residual Smoothing - effect of ε on convergence history of ONERA M6 wing. $M_\infty = 0.84, \alpha = 3.06°$, JITER=2, CFL=4.



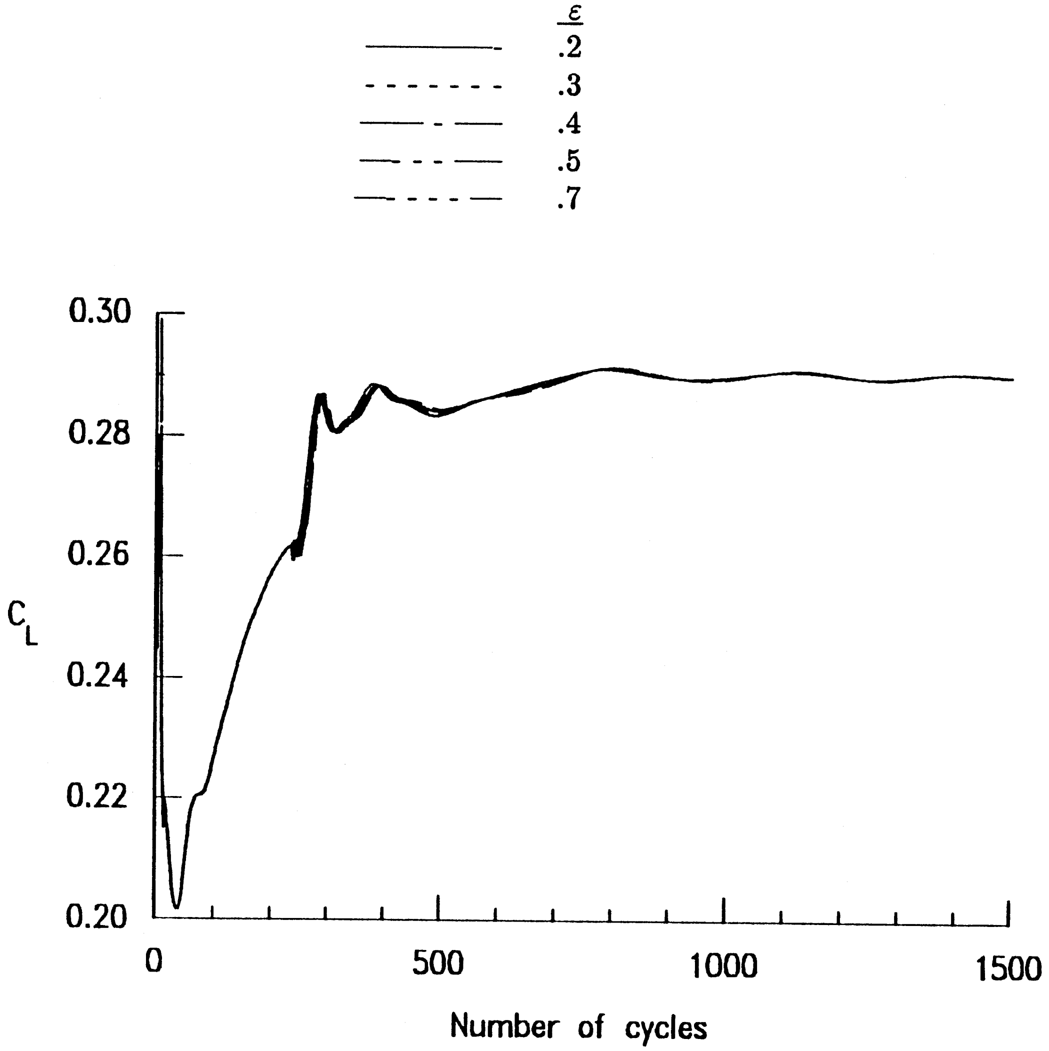
(b) L_2-norm vs. Iteration

Figure 12.- Continued.



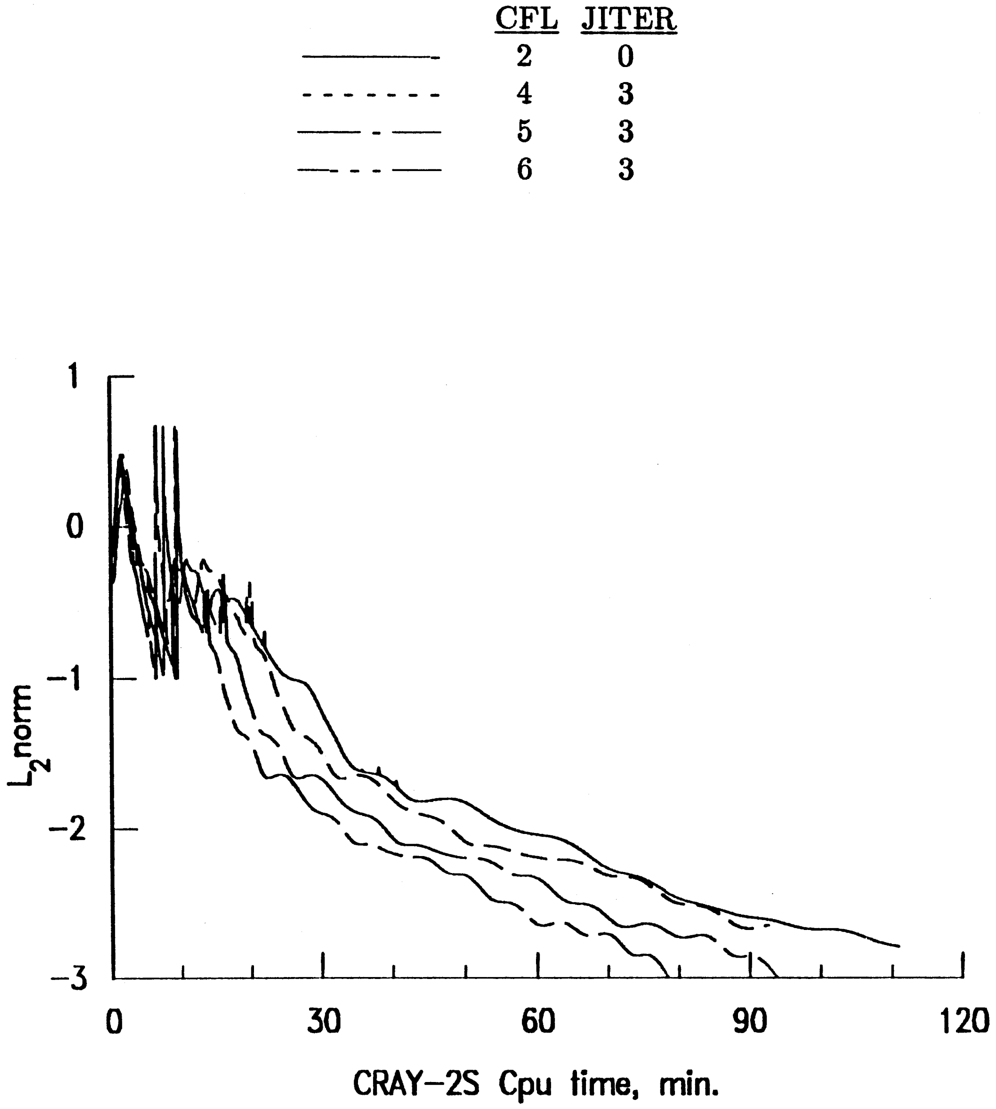
(c) Lift coefficient vs. CPU time

Figure 12.- Continued.



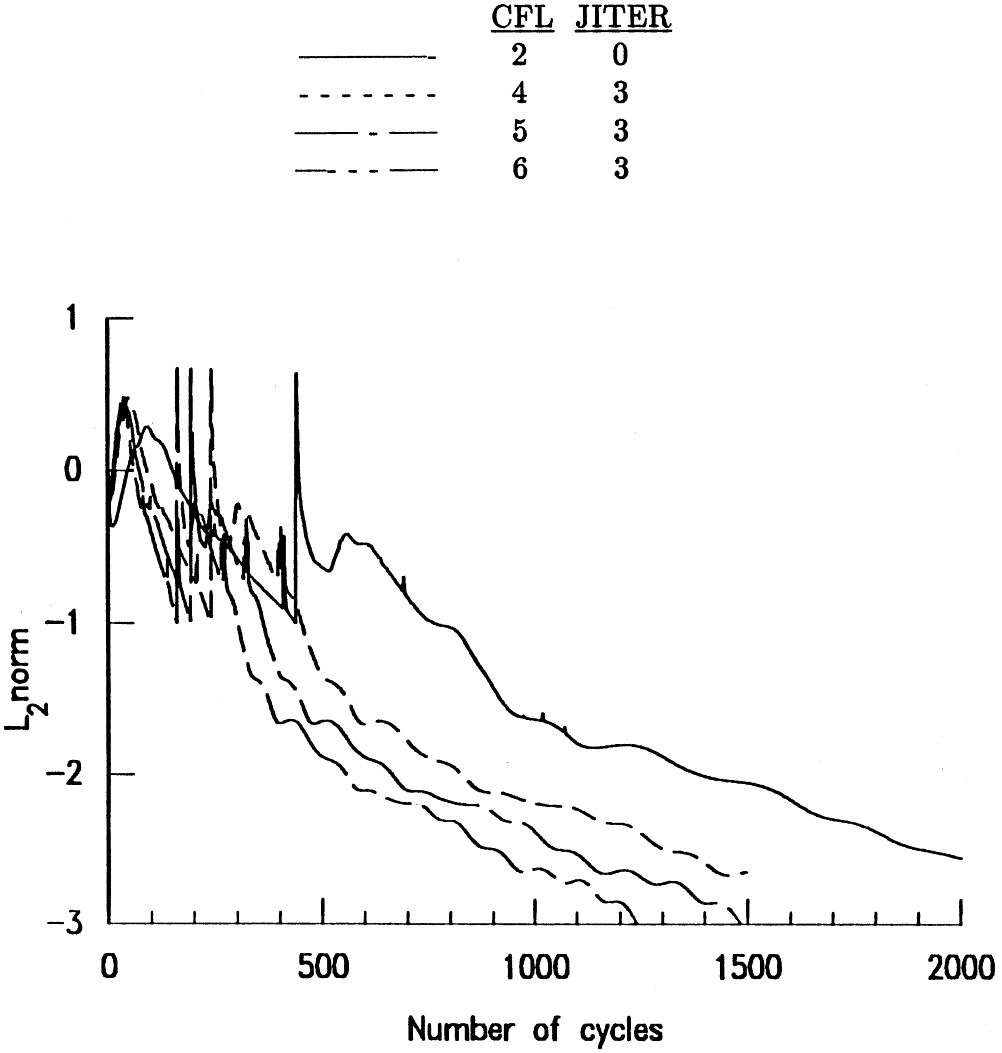
(d) Lift coefficient vs. Iteration

Figure 12.- Concluded.



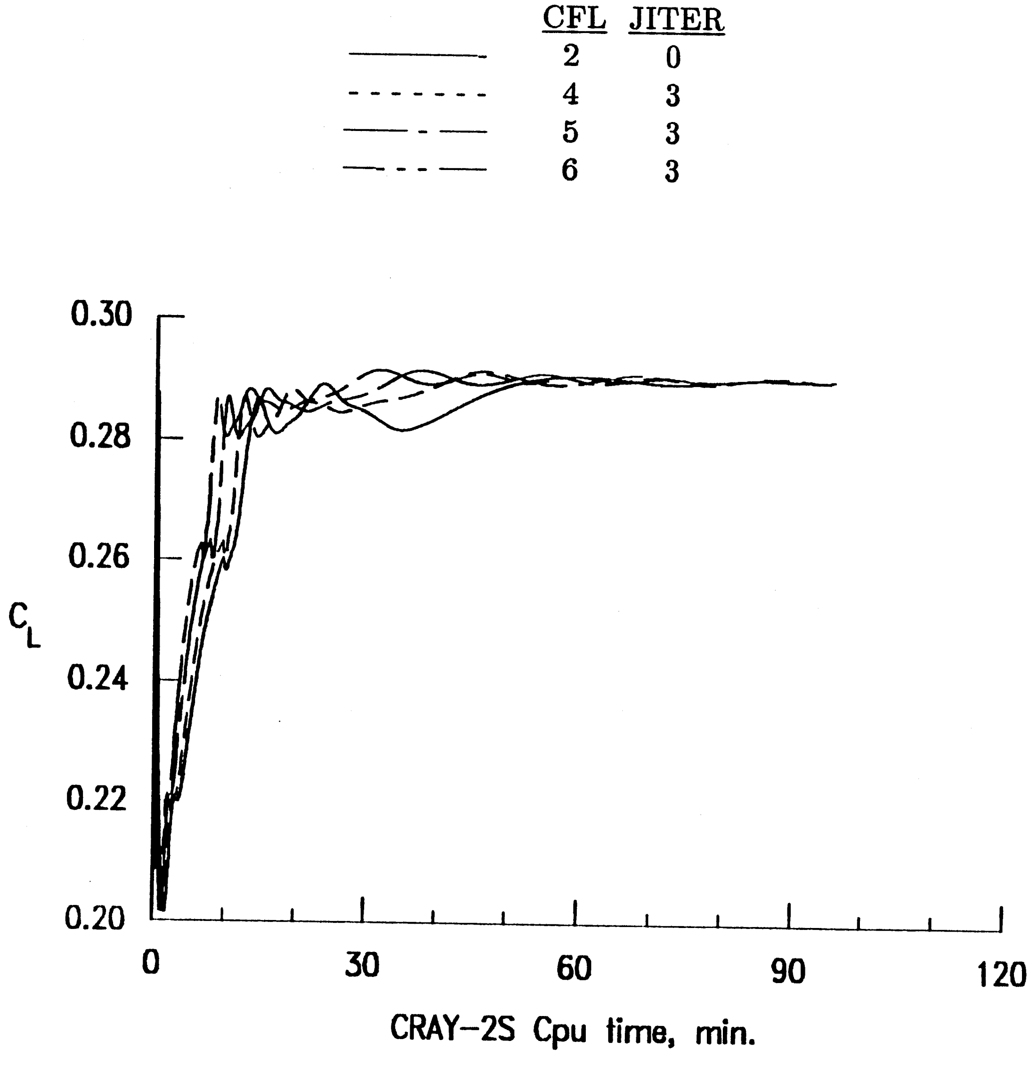
(a) L_2-norm vs. CPU time

Figure 13.- Implicit Residual Smoothing - effect of CFL on convergence history of ONERA M6 wing. $M_\infty = 0.84, \alpha = 3.06°, \varepsilon = 0.5$.



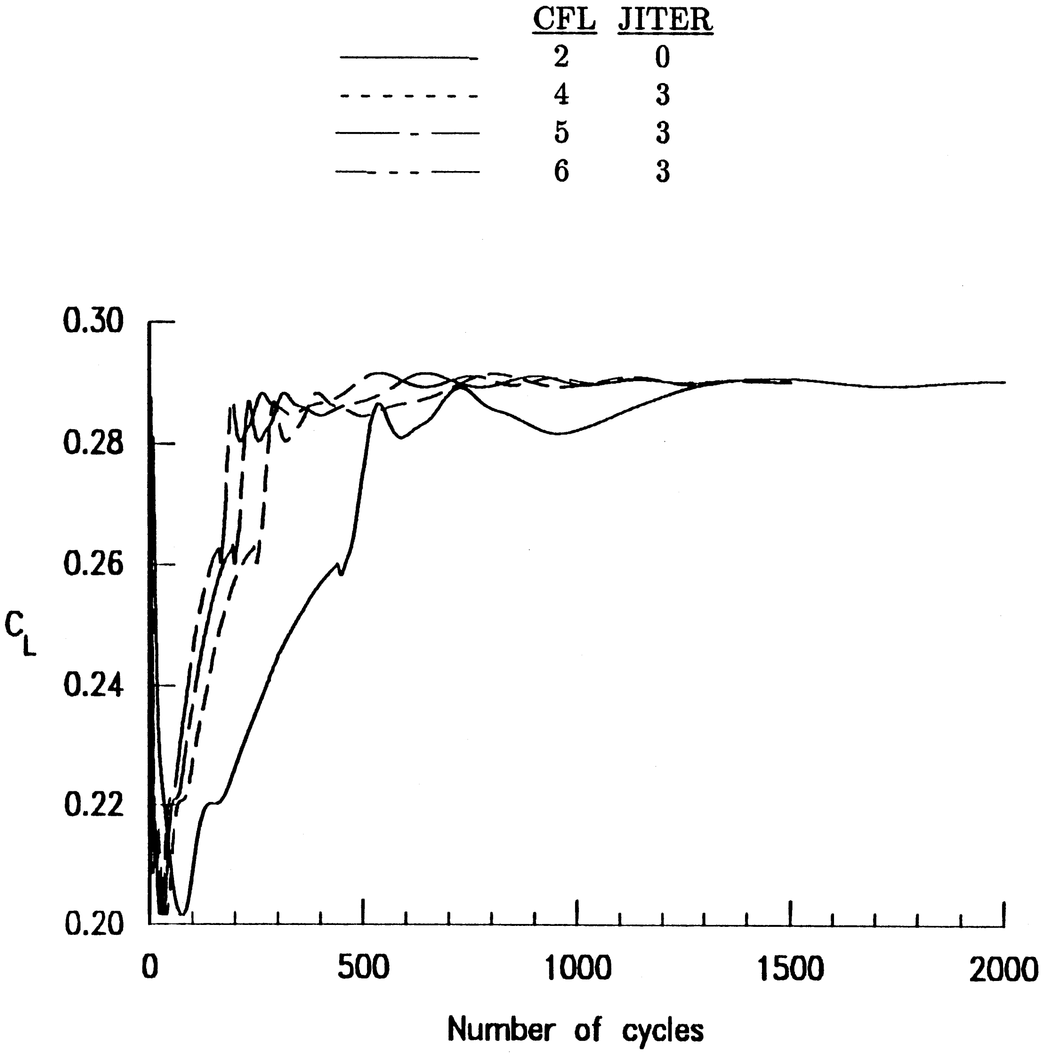
(b) L_2-norm vs. Iteration

Figure 13.- Continued.



(c) Lift coefficient vs. CPU time

Figure 13.- Continued.



(d) Lift coefficient vs. Iteration

Figure 13.- Concluded.

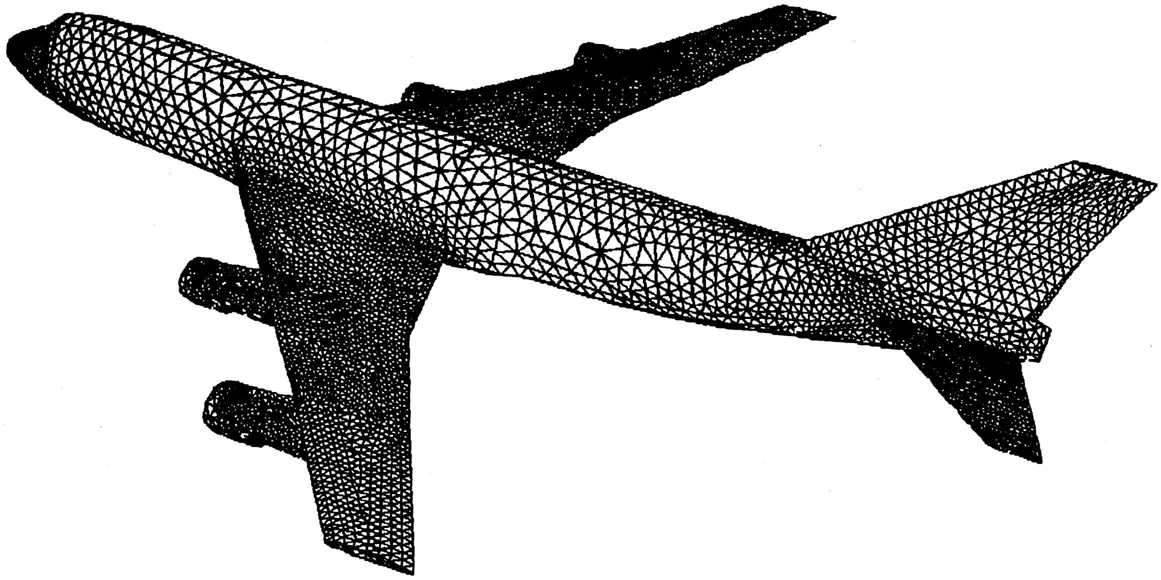


Figure 14.- Surface grid for Boeing 747-200 Configuration.

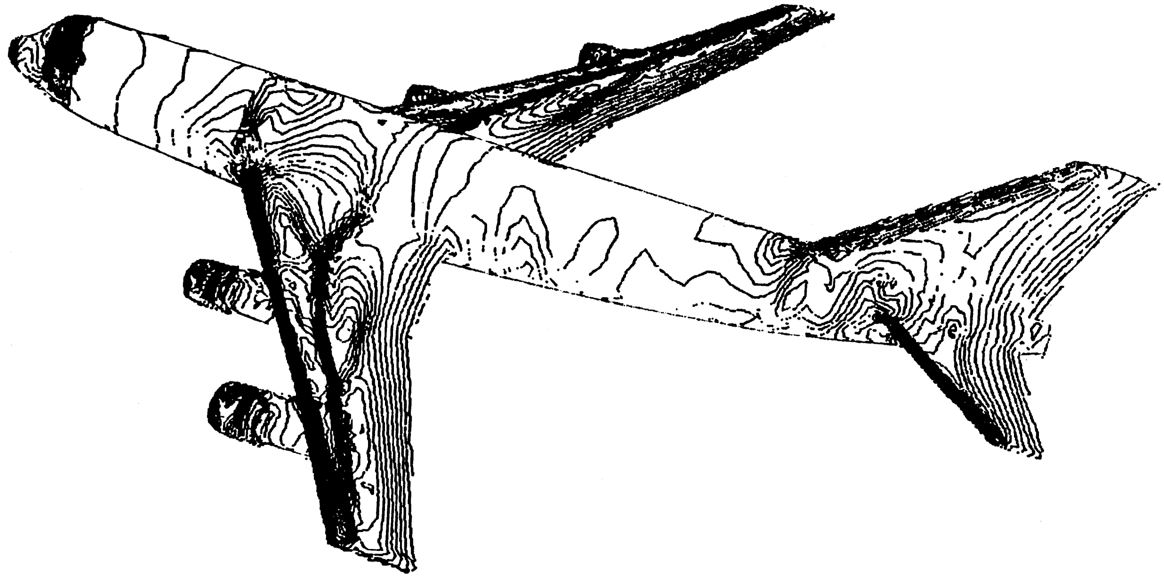
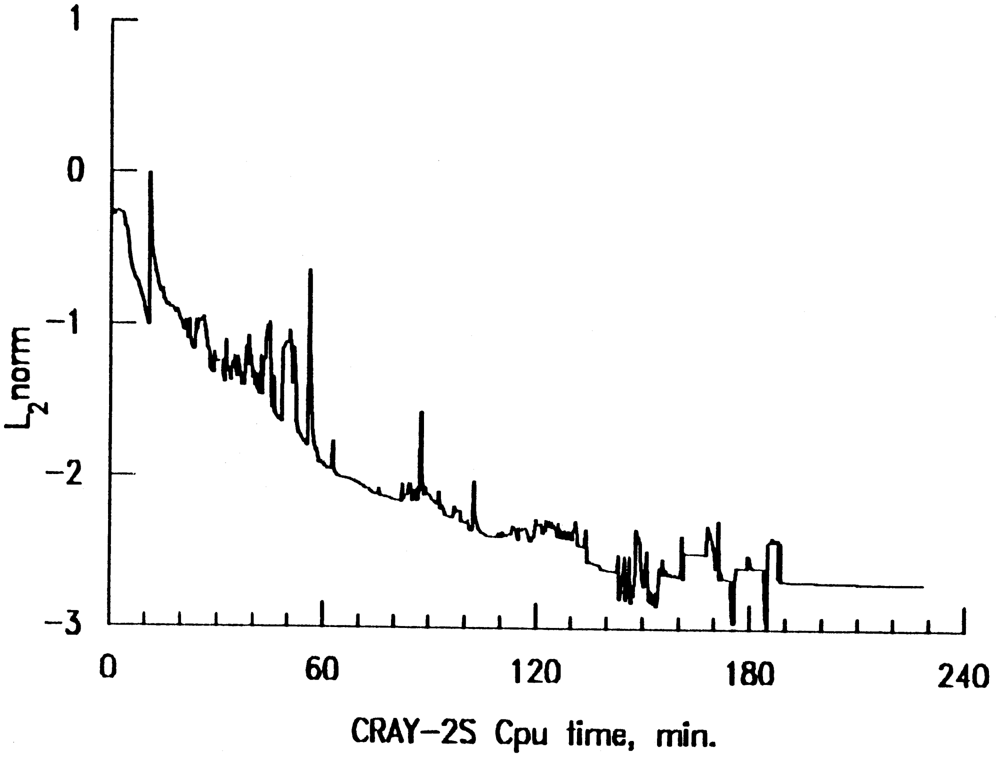
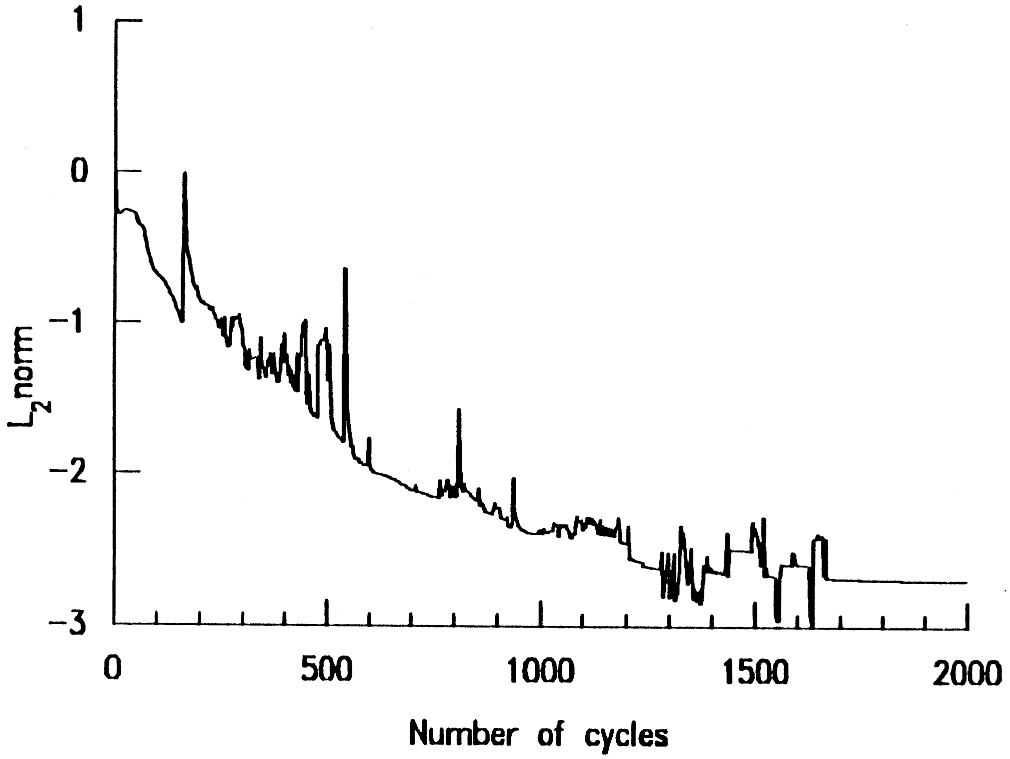


Figure 15.- Surface pressure contours for Boeing 747-200 Configuration. $M_\infty = 0.84, \alpha = 2.73^\circ, \Delta(p/p_\infty) = 0.02$.



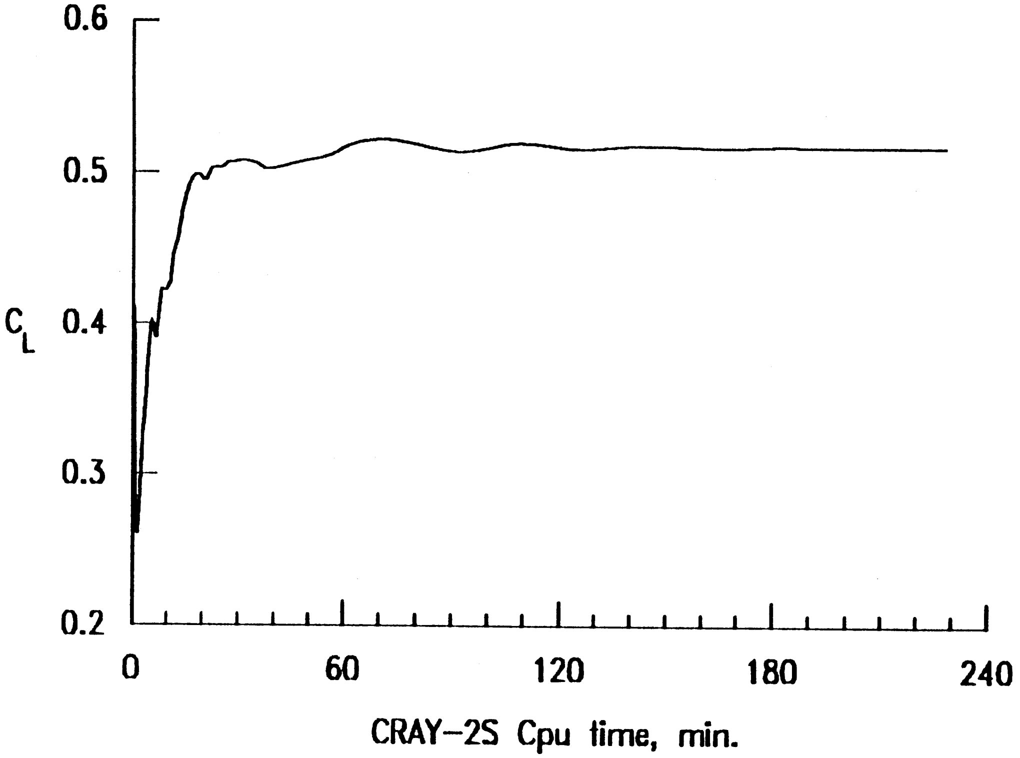
(a) L2-norm vs. CPU time

Figure 16.- Convergence history for Boeing 747-200.
$M_\infty = 0.84, \alpha = 3.06°, \varepsilon = 0.5$.



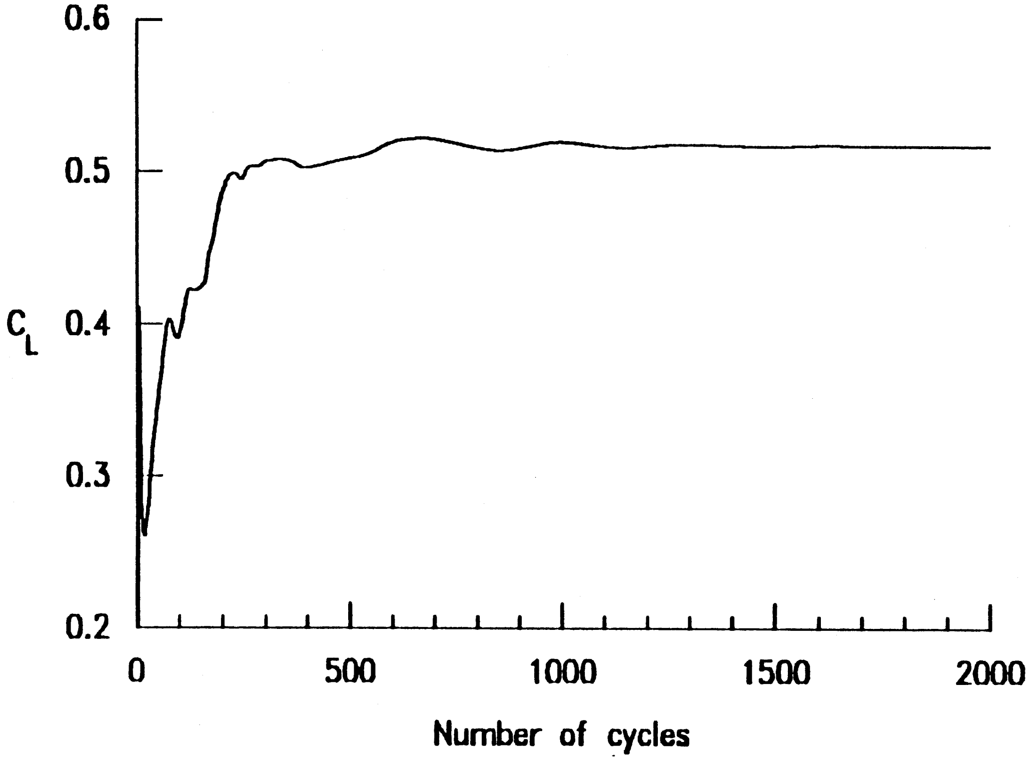
(b) L_2-norm vs. Iteration

Figure 16.- Continued.



(c) Lift coefficient vs. CPU time

Figure 16.- Continued.



(d) Lift coefficient vs. Iteration

Figure 16.- Concluded.

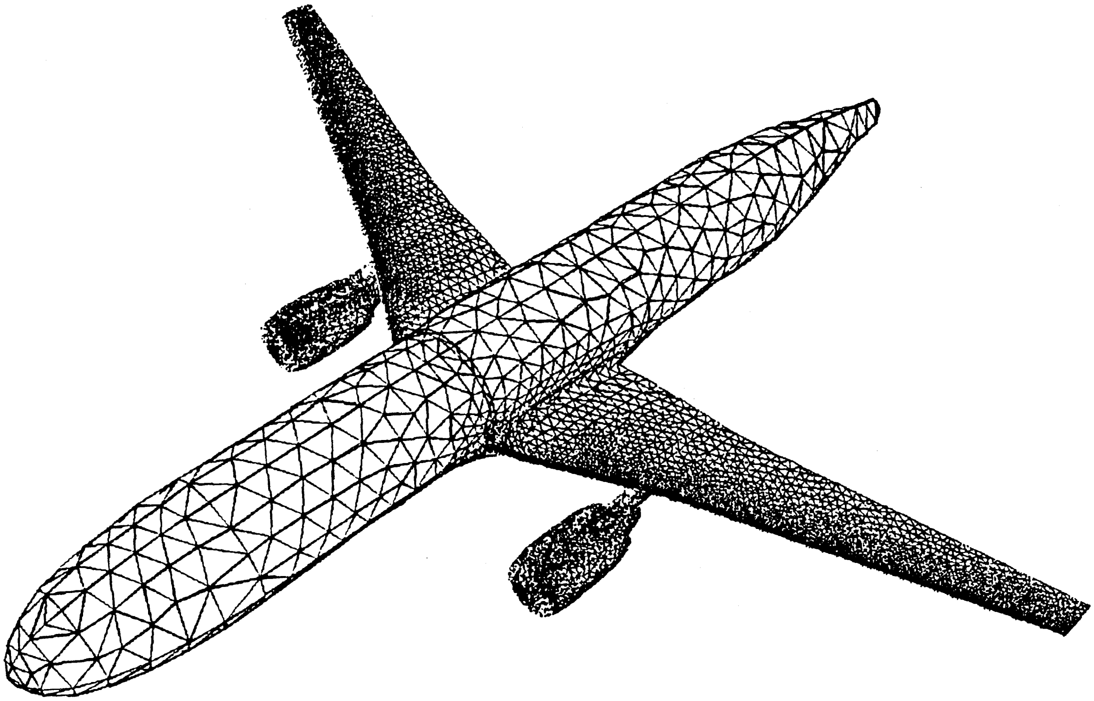


Figure 17.- Surface grid for Low-Wing Transport.

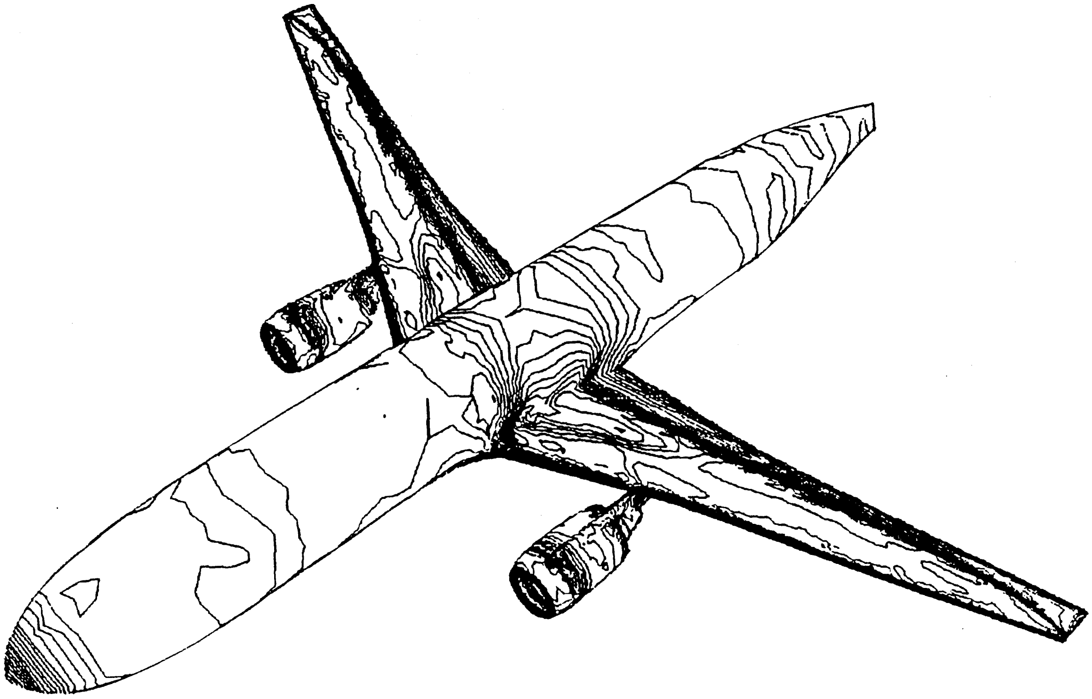


Figure 18.- Surface pressure contours for Low-Wing Transport.
Low-Wing Transport, $M_\infty = 0.768, \alpha = 1.116°$.

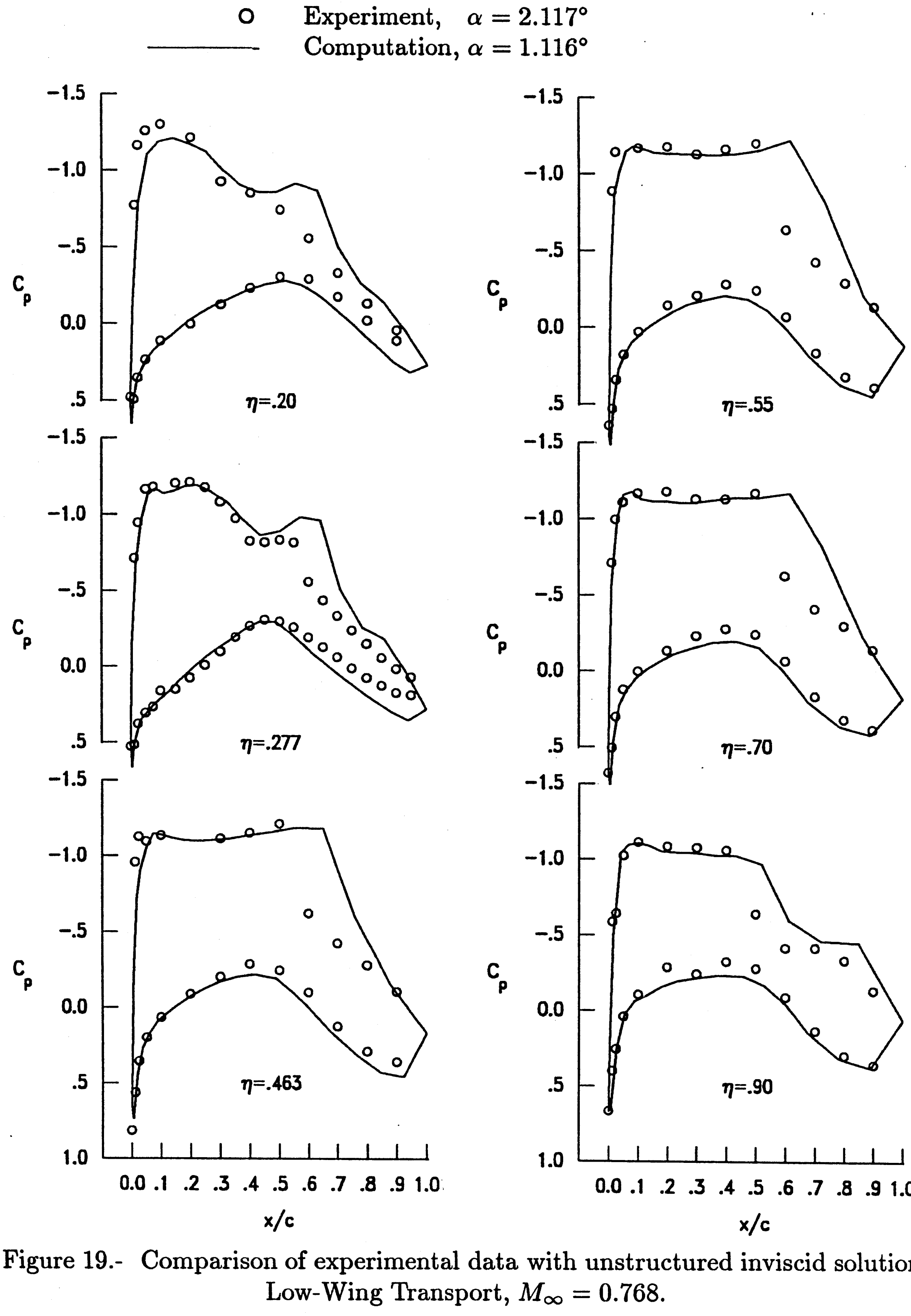


Figure 19.- Comparison of experimental data with unstructured inviscid solution. Low-Wing Transport, $M_\infty = 0.768$.

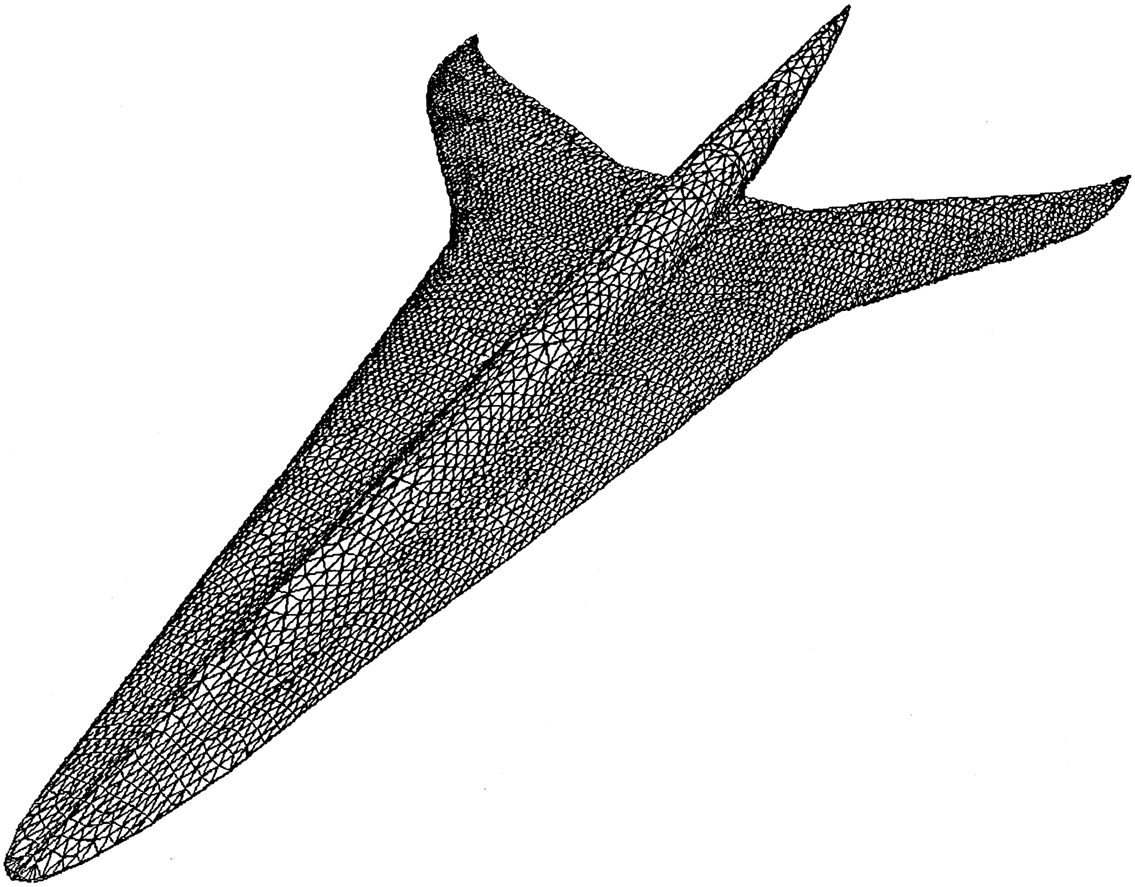


Figure 20.- Surface grid for High-Speed Civil Transport Configuration.

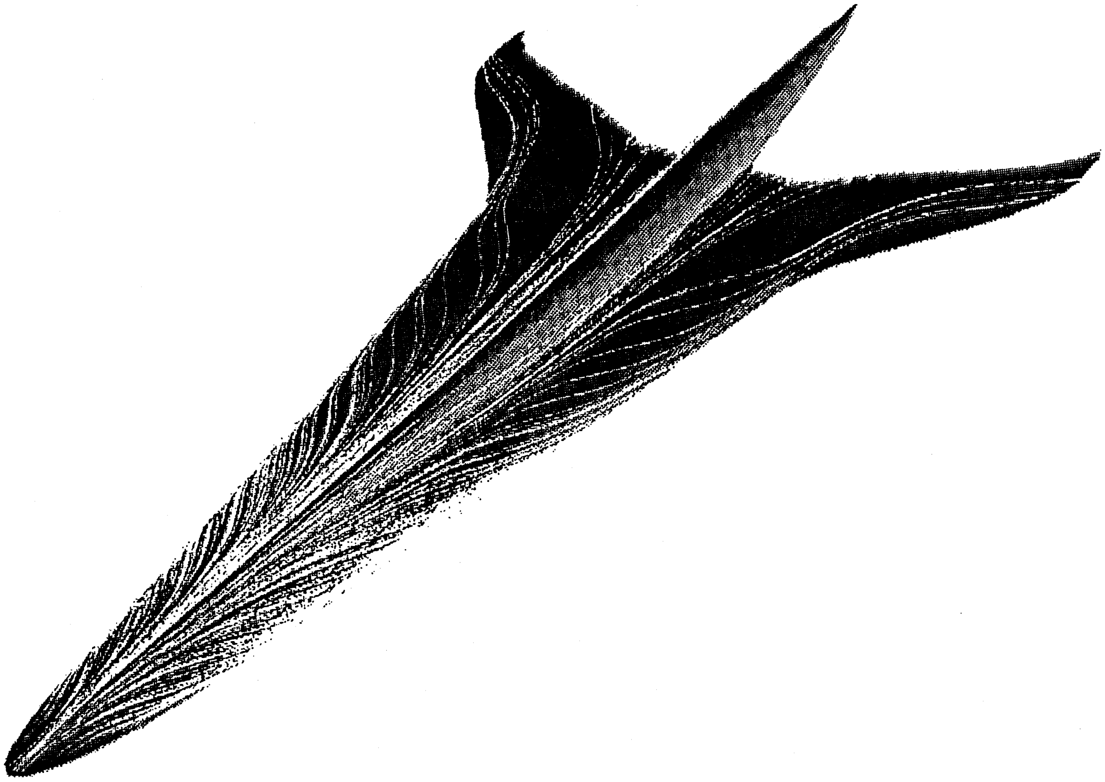
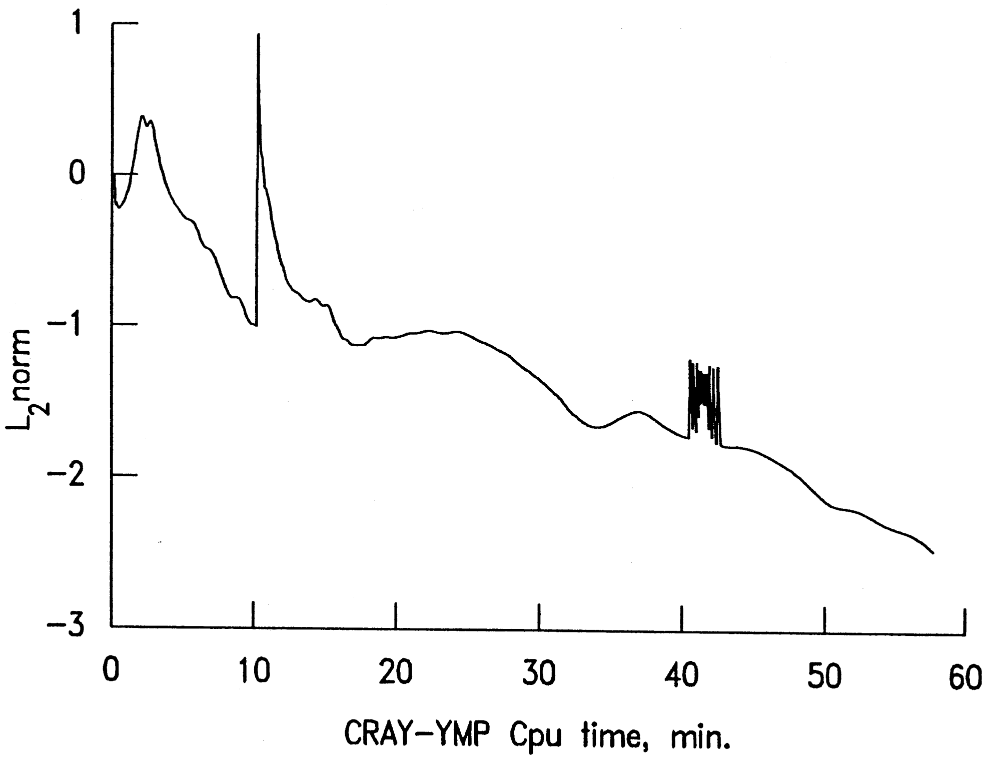
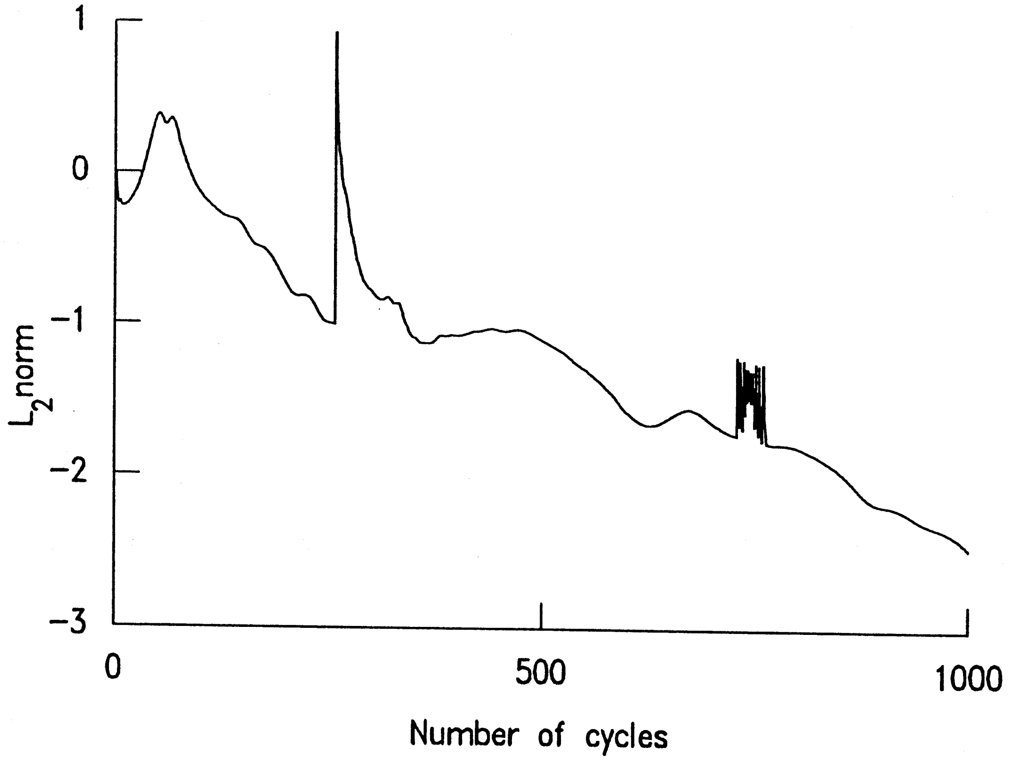


Figure 21.- Surface oilflow lines for High-Speed Civil Transport Configuration. $M_\infty = 0.90, \alpha = 6.47°$.



(a) L_2-norm vs. CPU time

Figure 22.- Convergence history for High-Speed Civil Transport Configuration. $M_\infty = 0.90, \alpha = 6.47°$.



(b) L_2-norm vs. Iteration

Figure 22.- Continued.

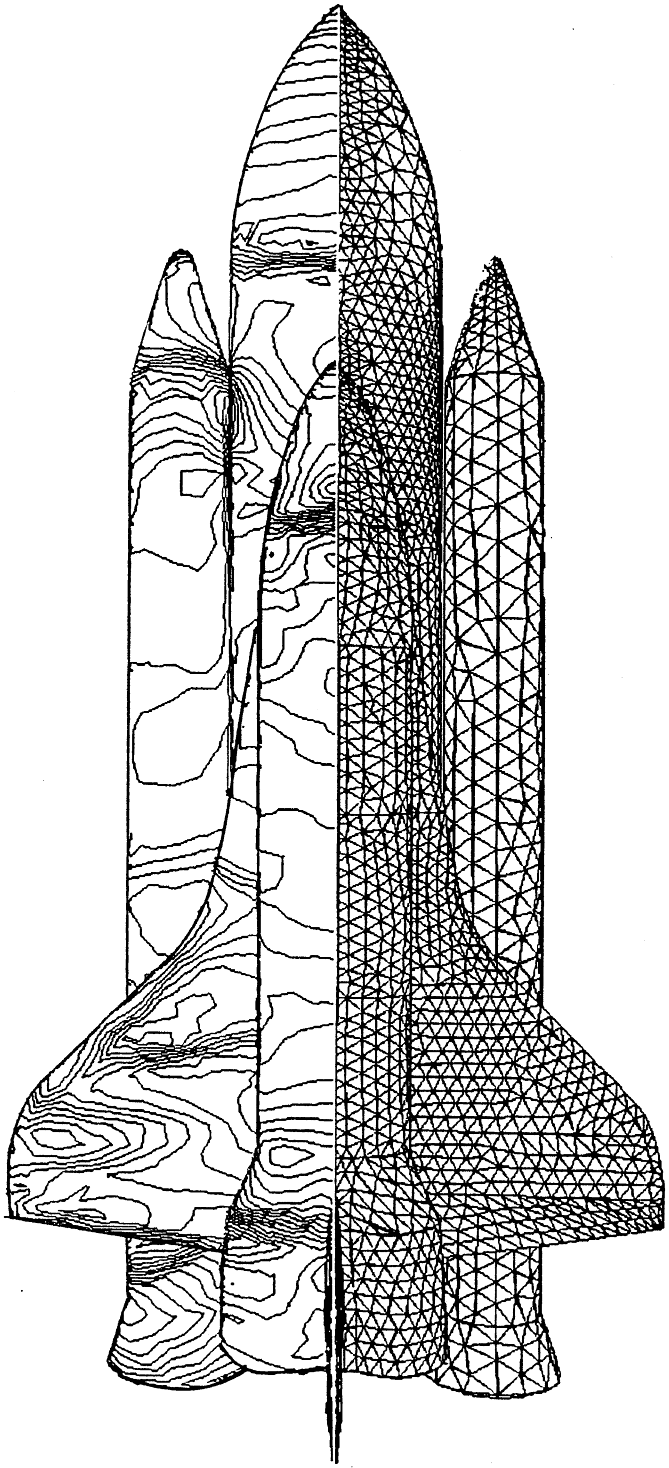


Figure 23.- Surface grid and pressure contours on STS.
$M_\infty = 1.05, \alpha = -3.1°$.

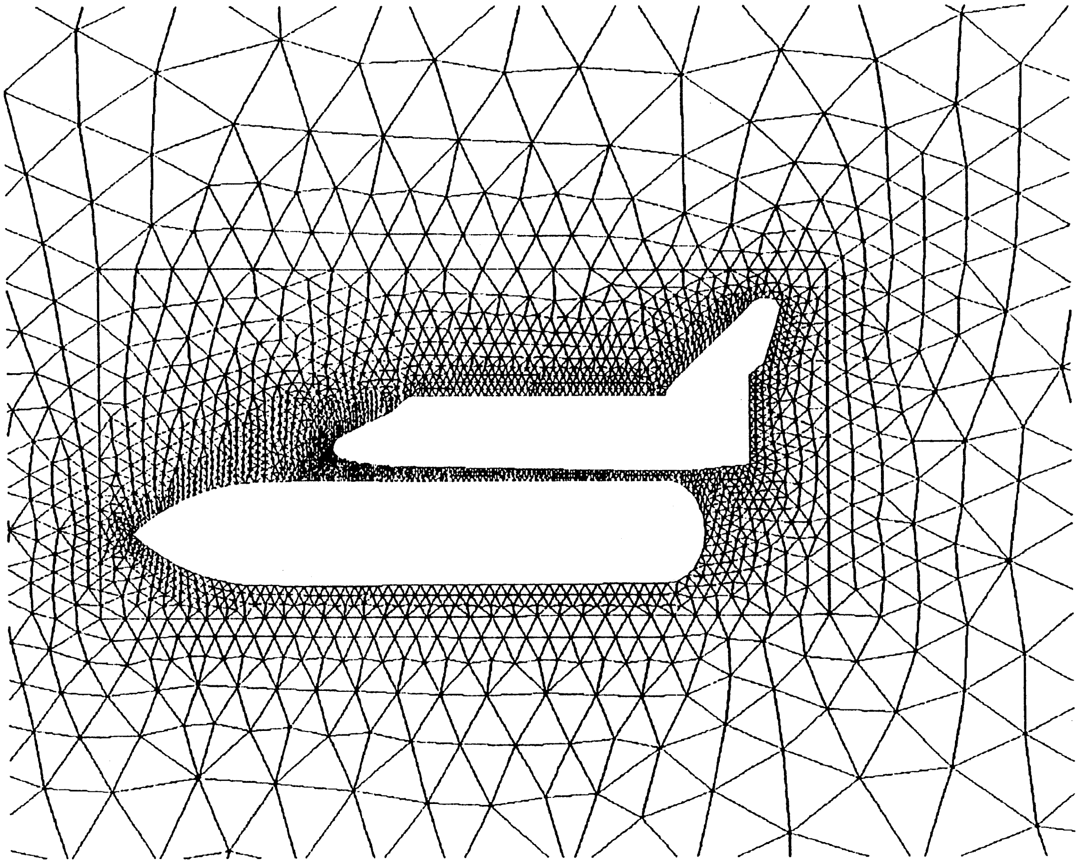


Figure 24.- Grid on symmetry plane for STS.

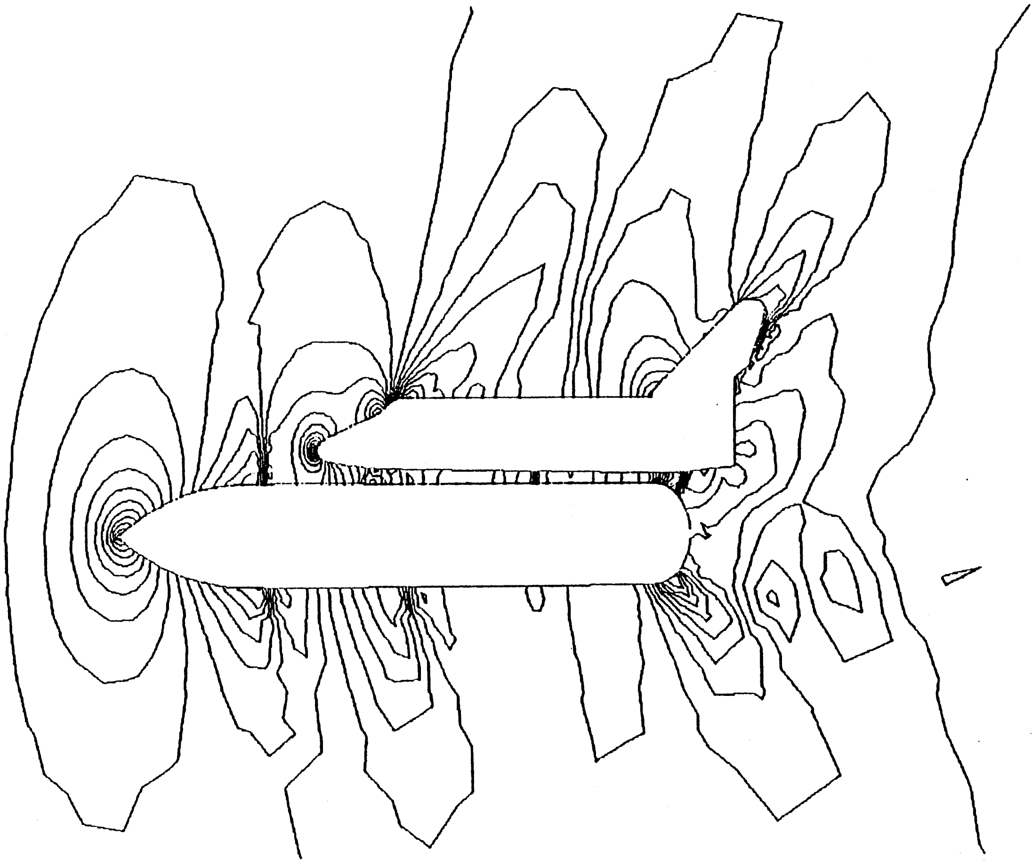
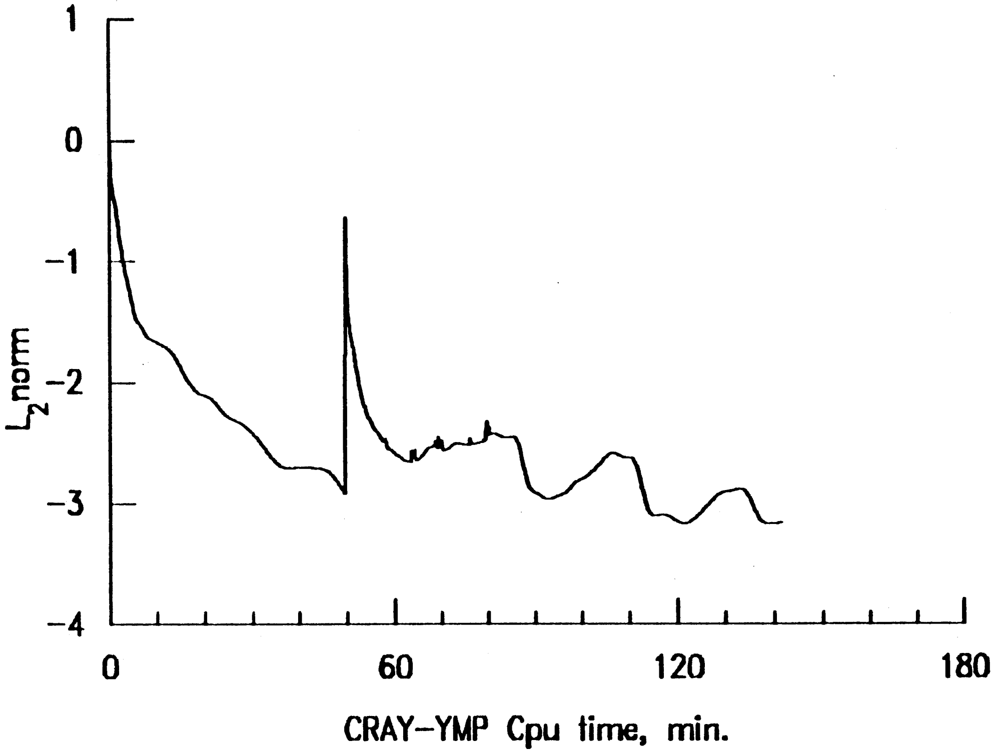
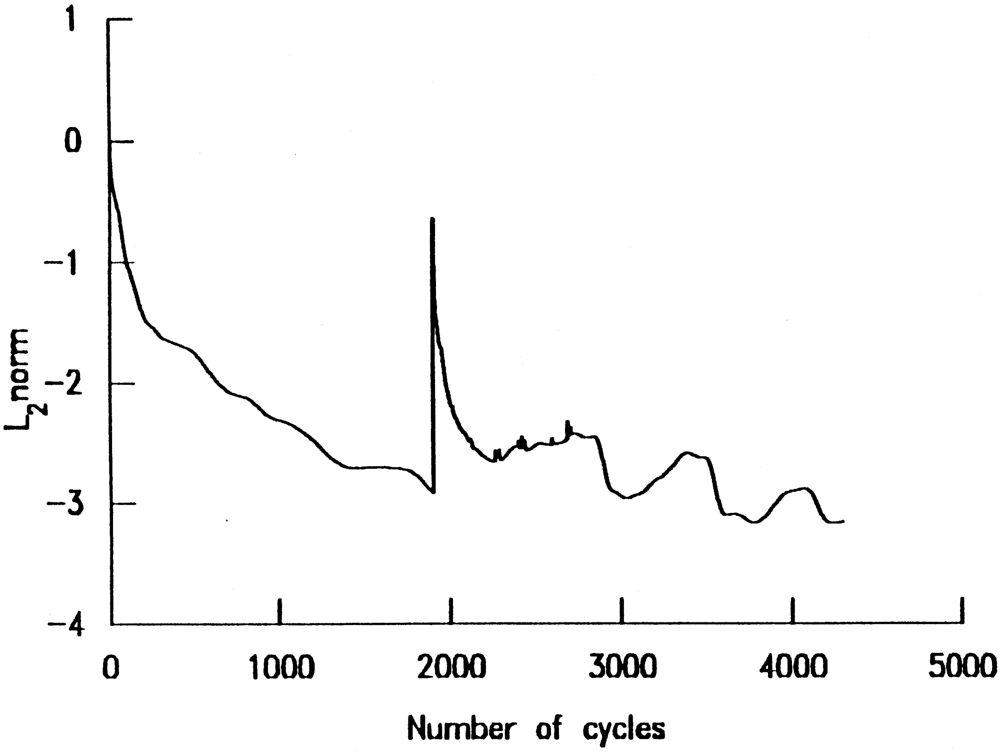


Figure 25.- Pressure contours on symmetry plane for STS.
$M_\infty = 1.05, \alpha = -3.1°$.



(a) L_2-norm vs. CPU time

Figure 26.- Convergence history for STS.
$M_\infty = 1.05, \alpha = -3.1°$.



(b) L_2-norm vs. Iteration

Figure 26.- Continued.

APPENDICES

A1 - DERIVATION OF METRIC TERMS FOR TETRAHEDRAL CELLS

The expressions for computing the metric terms for tetrahedral cells are presented in the following. Figure 3 should be used for interpreting the following relations.

Volume Centroid

The volume centroid of any tetrahedral cell is determined by the coordinate average of the four vertices:

$$\begin{aligned} x_c &= {}^1\!/_4(x_{n_1} + x_{n_2} + x_{n_3} + x_{n_4}) \\ y_c &= {}^1\!/_4(y_{n_1} + y_{n_2} + y_{n_3} + y_{n_4}) \\ z_c &= {}^1\!/_4(z_{n_1} + z_{n_2} + z_{n_3} + z_{n_4}). \end{aligned} \tag{A1.1}$$

Face Centroid

Similarly, the centroid of a triangular cell face is the coordinate average of the three vertices:

$$\begin{aligned} x_{f_{1,2,3}} &= {}^1\!/_3(x_{n_1} + x_{n_2} + x_{n_3}) \\ y_{f_{1,2,3}} &= {}^1\!/_3(y_{n_1} + y_{n_2} + y_{n_3}) \\ z_{f_{1,2,3}} &= {}^1\!/_3(z_{n_1} + z_{n_2} + z_{n_3}). \end{aligned} \tag{A1.2}$$

Directed Face Area

Consider face 1,2,3 and node 4 (ND4) of cell NC1 in Fig. 3. Let

$$\begin{aligned} \Delta x_2 &= x_2 - x_1, \quad \Delta y_2 = y_2 - y_1, \quad \Delta z_2 = z_2 - z_1 \\ \Delta x_3 &= x_3 - x_1, \quad \Delta y_3 = y_3 - y_1, \quad \Delta z_3 = z_3 - z_1 \\ \Delta x_4 &= x_4 - x_1, \quad \Delta y_4 = y_4 - y_1, \quad \Delta z_4 = z_4 - z_1. \end{aligned}$$

By defining the edges as vectors

$$\mathbf{L}_2 = \Delta x_2 \hat{\mathbf{i}} + \Delta y_2 \hat{\mathbf{j}} + \Delta z_2 \hat{\mathbf{k}}$$

$$\mathbf{L}_3 = \Delta x_3 \hat{\mathbf{i}} + \Delta y_3 \hat{\mathbf{j}} + \Delta z_3 \hat{\mathbf{k}}$$

$$\mathbf{L}_4 = \Delta x_4 \hat{\mathbf{i}} + \Delta y_4 \hat{\mathbf{j}} + \Delta z_4 \hat{\mathbf{k}}$$

then area of the triangular face can be computed as

$$\begin{aligned} A_{f_{1,2,3}} = {}^1\!/_2 \mid \mathbf{L}_2 \times \mathbf{L}_3 \mid = {}^1\!/_2[(\Delta y_2 \Delta z_3 - \Delta z_2 \Delta y_3)^2 \\ +(\Delta x_3 \Delta z_2 - \Delta x_2 \Delta z_3)^2 \\ +(\Delta x_2 \Delta y_3 - \Delta x_3 \Delta y_2)^2]^{1/2} \end{aligned} \qquad (A1.3)$$

with the direction cosines defined by

$$\hat{\mathbf{n}}_{f_{1,2,3}} = \frac{\mathbf{L}_2 \times \mathbf{L}_3}{\mid \mathbf{L}_2 \times \mathbf{L}_3 \mid} = n_x \hat{\mathbf{i}} + n_y \hat{\mathbf{j}} + n_z \hat{\mathbf{k}} \qquad (A1.4)$$

where

$$\begin{aligned} n_x &= \frac{1}{2A_{f_{1,2,3}}}(\Delta y_2 \Delta z_3 - \Delta z_2 \Delta y_3) \\ n_y &= \frac{1}{2A_{f_{1,2,3}}}(\Delta x_3 \Delta z_2 - \Delta x_2 \Delta z_3) \\ n_y &= \frac{1}{2A_{f_{1,2,3}}}(\Delta x_2 \Delta y_3 - \Delta x_3 \Delta y_2). \end{aligned}$$

The unit normal is forced to point outward to cell NC1 in Fig. 3 by convention. Once the grid and connectivity files have been generated by a grid generator, a check is made to insure that the face normals satisfy this condition. This is accomplished by making the check

$$if\ (\hat{\mathbf{n}}_{f_{1,2,3}} \cdot \mathbf{L}_4)_{NC1} > 0$$

then switch nodes 2 and 3 by redefining

$$n_x = -n_x, \quad n_y = -n_y, \quad n_z = -n_z$$

and setting IFACE(NF,4)=ND3 and IFACE(NF,5)=ND2 in the face connectivity array.

Cell Volume

Tetrahedral cell volume is computed by the vector relation

$$\begin{aligned} V &= -{}^1\!/_6[(\mathbf{L}_2 \times \mathbf{L}_3) \cdot \mathbf{L}_4] \\ &= -{}^1\!/_6(2A_{f_{1,2,3}})[\hat{\mathbf{n}}_{f_{1,2,3}} \cdot \mathbf{L}_4] \\ &= {}^1\!/_3 A_{f_{1,2,3}}(n_x \Delta x_4 + n_y \Delta y_4 + n_z \Delta z_4). \end{aligned} \qquad (A1.5)$$

Projected Cell Area

The projected cell area is use in the relation for local time stepping, Eq. (4.3). Consider an arbitrary tetrahedral cell, c, in space, e.g. cell NC1 in Fig. 3. The four outward pointing face-normal vectors are

$$\begin{aligned}
\hat{\mathbf{n}}_{123} &= n_{x_{123}}\hat{\mathbf{i}} + n_{y_{123}}\hat{\mathbf{j}} + n_{z_{123}}\hat{\mathbf{k}} \\
\hat{\mathbf{n}}_{142} &= n_{x_{142}}\hat{\mathbf{i}} + n_{y_{142}}\hat{\mathbf{j}} + n_{z_{142}}\hat{\mathbf{k}} \\
\hat{\mathbf{n}}_{134} &= n_{x_{134}}\hat{\mathbf{i}} + n_{y_{134}}\hat{\mathbf{j}} + n_{z_{134}}\hat{\mathbf{k}} \\
\hat{\mathbf{n}}_{243} &= n_{x_{243}}\hat{\mathbf{i}} + n_{y_{243}}\hat{\mathbf{j}} + n_{z_{243}}\hat{\mathbf{k}}.
\end{aligned}$$

The projected area of cell i can be computed by

$$\begin{aligned}
A_i^{(x)} &= {}^1\!/_2\sum_{j=1}^{\kappa(i)}(\hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{i}}) + \mid \hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{i}} \mid \\
A_i^{(y)} &= {}^1\!/_2\sum_{j=1}^{\kappa(i)}(\hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{j}}) + \mid \hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{j}} \mid \qquad (A1.6) \\
A_i^{(z)} &= {}^1\!/_2\sum_{j=1}^{\kappa(i)}(\hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{k}}) + \mid \hat{\mathbf{n}}A_{f_{i,j}} \cdot \hat{\mathbf{k}} \mid .
\end{aligned}$$

A2 - DERIVATION OF SIMPLIFIED HIGHER-ORDER SCHEME

In the following, a universal expression for the Taylor series expansion within a triangular cell will be derived. A similar derivation could be carried out for a tetrahedral cell. The resulting expression requires knowledge of the state at the vertices or nodes, which is obtained by a weighted average from the state at the centers of the cells surrounding the node.

Start with a Taylor series expansion of the primative variable in the neighborhood of each cell center (x_c, y_c, z_c):

$$\begin{aligned} \mathbf{q}(x,y,z) &= \mathbf{q}(x_c,y_c,z_c) + \nabla\mathbf{q}_c \cdot \mathbf{\Delta r} \quad + \quad O(\Delta r^2) \\ &\approx \mathbf{q}(x_c,y_c,z_c) + \mathbf{q}_x|_c \cdot (x - x_c) \qquad\qquad (A2.1) \\ &\quad + \mathbf{q}_y|_c \cdot (y - y_c) \end{aligned}$$

where

$$\mathbf{q} \equiv [\rho, u, v, w, p]^T$$

This formulation requires gradient information at the cell centers. The gradients are computed from an approximate form of the exact relation:

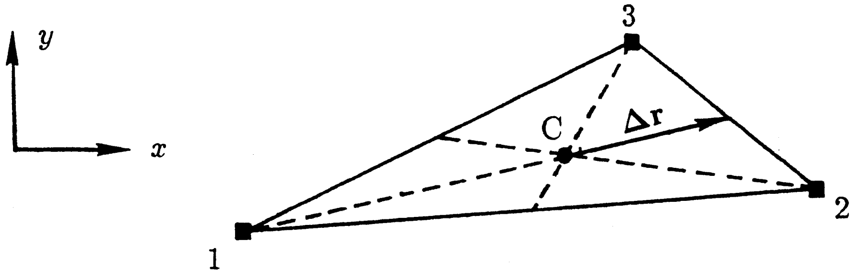
$$\int\int_{\Omega} \nabla\mathbf{q}\,dA = \oint_{\partial\Omega} \mathbf{q}\hat{\mathbf{n}}\,dl \qquad (A2.2)$$

The averaged solution gradient is estimated by computing the closed integral of Eq. (A2.2) over the boundary $\partial\Omega$ enclosing the domain Ω

$$\nabla\mathbf{q}_c = \frac{1}{A_\Omega} \oint_{\partial\Omega} \mathbf{q}\hat{\mathbf{n}}\,dl \qquad (A2.3)$$

In two-dimensions, Eq. (A2.3) represents a line integral around some closed path surrounding an area. In three-dimensions, it represents a surface integral over a surface enclosing a volume. The unit normal $\hat{\mathbf{n}}$ is assumed to point outward from the domain.

The components of the gradient in Eq. (A2.1) can be computed as in Ref. [43] by approximating the solution to Eq. (A2.3) with the midpoint trapezoidal rule. Consider the arbitrary triangular cell in sketch i.



Sketch i

The goal is to expand the solution from the cell centroid to the centroids of the edges, e.g. to edge 2-3. Note that the vector $\Delta\mathbf{r}$ is defined along the line passing from node 1 through the cell centroid to the centroid of the opposite edge 2-3. When the midpoint trapezoidal rule is applied to the integral in Eq. (A2.3) around the perimeter of the triangle, then the x-component is written as

$$\begin{aligned} \mathbf{q}_x|_c &= \frac{1}{2A}\left[(\mathbf{q}_{n_1}+\mathbf{q}_{n_2})n_{x_{12}}l_{12}+(\mathbf{q}_{n_2}+\mathbf{q}_{n_3})n_{x_{23}}l_{23}+(\mathbf{q}_{n_3}+\mathbf{q}_{n_1})n_{x_{31}}l_{31}\right] \\ &= \frac{1}{2A}[(\mathbf{q}_{n_1}+\mathbf{q}_{n_2})(y_2-y_1)+(\mathbf{q}_{n_2}+\mathbf{q}_{n_3})(y_3-y_2)+(\mathbf{q}_{n_3}+\mathbf{q}_{n_1})(y_1-y_3)] \end{aligned} \tag{A2.4}$$

where A is the area of the triangle, l_{12}, l_{23}, l_{31} are edge lengths and $n_{x_{12}}$, $n_{x_{23}}$, $n_{x_{31}}$ the x-component of the direction cosines for edges 1-2, 2-3, and 3-1. To further simplify, define

$$\langle\mathbf{q}_{12}\rangle = {}^1\!/\!_2(\mathbf{q}_{n_1}+\mathbf{q}_{n_2}),\quad \langle\mathbf{q}_{23}\rangle = {}^1\!/\!_2(\mathbf{q}_{n_2}+\mathbf{q}_{n_3}),\quad \langle\mathbf{q}_{31}\rangle = {}^1\!/\!_2(\mathbf{q}_{n_3}+\mathbf{q}_{n_1}) \tag{A2.5}$$

and

$$\begin{aligned} \Delta x_{12} &= x_2-x_1, \quad \Delta x_{23} = x_3-x_2, \quad \Delta x_{31} = x_1-x_3 \\ \Delta y_{12} &= y_2-y_1, \quad \Delta y_{23} = y_3-y_2, \quad \Delta y_{31} = y_1-y_3 \end{aligned} \tag{A2.6}$$

Rewrite Eq. (A2.4) as:

$$\mathbf{q}_x = \frac{1}{A}\left[\langle\mathbf{q}_{12}\rangle\Delta y_{12}+\langle\mathbf{q}_{23}\rangle\Delta y_{23}+\langle\mathbf{q}_{31}\rangle\Delta y_{31}\right] \tag{A2.7}$$

Similarly,

$$\mathbf{q}_y = \frac{-1}{A}\left[\langle \mathbf{q}_{12}\rangle \Delta x_{12} + \langle \mathbf{q}_{23}\rangle \Delta x_{23} + \langle \mathbf{q}_{31}\rangle \Delta x_{31}\right] \tag{A2.8}$$

As shown in sketch i,

$$\begin{aligned} \mathbf{\Delta r} &= {}^1\!/\!_3\{[{}^1\!/\!_2(x_2 + x_3) - x_1]\hat{\mathbf{i}} + [{}^1\!/\!_2(y_2 + y_3) - y_1]\hat{\mathbf{j}}\} \\ &= {}^1\!/\!_6[(\Delta x_{12} - \Delta x_{31})\hat{\mathbf{i}} \; + \; (\Delta y_{12} - \Delta y_{31})\hat{\mathbf{j}}] \end{aligned} \tag{A2.9}$$

Equation (A2.9) applies to any arbitrary triangle due to the invariant features that 1) a line extending from a vertex through the cell centroid will always intersect the centroid of the opposing edge, and 2) the distance from the vertex to the centroid of the triangle is always two-thirds of that from the vertex to the opposing edge. Since $\nabla \mathbf{q} = \mathbf{q}_x\hat{\mathbf{i}} + \mathbf{q}_y\hat{\mathbf{j}}$ the first order term in Eq. (A2.1) becomes

$$\begin{aligned} \nabla \mathbf{q}_c \cdot \mathbf{\Delta r} = \frac{1}{6A}[(&\langle \mathbf{q}_{12}\rangle \Delta y_{12} + \langle \mathbf{q}_{23}\rangle \Delta y_{23} + \langle \mathbf{q}_{31}\rangle \Delta y_{31})(\Delta x_{12} - \Delta x_{31}) \\ &-(\langle \mathbf{q}_{12}\rangle \Delta x_{12} + \langle \mathbf{q}_{23}\rangle \Delta x_{23} + \langle \mathbf{q}_{31}\rangle \Delta x_{31})(\Delta y_{12} - \Delta y_{31})] \\ = \frac{1}{6A}[&\langle \mathbf{q}_{23}\rangle(\Delta x_{12}\Delta y_{23} - \Delta x_{23}\Delta y_{12}) + \langle \mathbf{q}_{23}\rangle(\Delta x_{23}\Delta y_{31} - \Delta x_{31}\Delta y_{23}) \\ &+ (\langle \mathbf{q}_{12}\rangle + \langle \mathbf{q}_{31}\rangle)(\Delta x_{12}\Delta y_{31} - \Delta x_{31}\Delta y_{12})] \end{aligned} \tag{A2.10}$$

By defining the edges as vectors

$$\mathbf{L}_{12} = \Delta x_{12}\hat{\mathbf{i}} + \Delta y_{12}\hat{\mathbf{j}}$$

$$\mathbf{L}_{23} = \Delta x_{23}\hat{\mathbf{i}} + \Delta y_{23}\hat{\mathbf{j}}$$

$$\mathbf{L}_{31} = \Delta x_{31}\hat{\mathbf{i}} + \Delta y_{31}\hat{\mathbf{j}}$$

then area of the triangle can be computed as

$$\begin{aligned} 2A = \mid \mathbf{L}_{12} \times \mathbf{L}_{23} \mid &= \Delta x_{12}\Delta y_{23} - \Delta x_{23}\Delta y_{12} \\ \mid \mathbf{L}_{23} \times \mathbf{L}_{31} \mid &= \Delta x_{23}\Delta y_{31} - \Delta x_{31}\Delta y_{23} \\ \mid \mathbf{L}_{31} \times \mathbf{L}_{12} \mid &= \Delta x_{31}\Delta y_{12} - \Delta x_{12}\Delta y_{31} \end{aligned} \tag{A2.11}$$

Substitute Eq. (A2.11) into Eq. (A2.10)

$$\begin{aligned}\nabla \mathbf{q} \cdot \Delta \mathbf{r} &= {}^1\!/\!_3[2\langle \mathbf{q}_{23}\rangle - (\langle \mathbf{q}_{12}\rangle + \langle \mathbf{q}_{31}\rangle)] \\ &= {}^1\!/\!_3[(\mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - {}^1\!/\!_2(\mathbf{q}_{n_1} + \mathbf{q}_{n_2} + \mathbf{q}_{n_3} + \mathbf{q}_{n_1})] \\ &= {}^1\!/\!_3[{}^1\!/\!_2(\mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - \mathbf{q}_{n_1}]\end{aligned} \qquad (A2.12)$$

Substituting Eq. (A2.12) into Eq. (A2.1) gives the simple relation

$$\mathbf{q}_{e_{2,3}} = \mathbf{q}_c + {}^1\!/\!_3\,[{}^1\!/\!_2(\mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - \mathbf{q}_{n_1}] \qquad \textbf{(A2.13)}$$

A similar derivation in three-dimensions on a tetrahedral cell yields the expression

$$\mathbf{q}_{f_{1,2,3}} = \mathbf{q}_c + {}^1\!/\!_4\,[{}^1\!/\!_3(\mathbf{q}_{n_1} + \mathbf{q}_{n_2} + \mathbf{q}_{n_3}) - \mathbf{q}_{n_4}] \qquad \textbf{(A2.14)}$$

Thus, the state at the edge or face centroid can be readily determined by Eq. (A2.13) in two dimensions and Eq. (A2.14) in three dimensions, respectively. Recall that the edge or face centroid locations are precisely where the averaged fluxes are evaluated in the finite-volume formulation.

The Eqs. (A2.13) and (A2.14) are formally second-order accurate. This is evident by observing that the Taylor series in Eq. (A2.1) is second-order accurate and that the midpoint trapezoidal rule used to evaluate the integral in Eq. (A2.3) is also second-order accurate. The only question related to overall accuracy arises from the averaging procedure used in determining the nodal states. The accuracy of this procedure has not been determined, thus the method is only referred to as "higher-order" accurate.

VITA

Neal T. Frink was born in Asheville, North Carolina on November 23, 1953. He received his high school degree in the same city from A. C. Reynolds High School in 1972. He attended North Carolina State University, Raleigh, North Carolina as an Aerospace Engineering major while supplementing his education through the cooperative education program with NASA Langley Research Center, Hampton, Virginia. He graduated with honors in 1977. This was followed by a Master of Science degree in Aerospace Engineering in 1979, also from North Carolina State University.

From 1980 to the present, he has been employed by NASA at the Langley Research Center. Through NASA's graduate training program, he attended Virginia Polytechnic Institute and State University from the fall of 1987 through the spring of 1988 to satisfy the academic requirements for a Doctor of Philosophy degree in Aerospace Engineering.

His past work experiences include both experimental wind-tunnel studies and applied computation research. His primary research interest is in applied computational fluid dynamics.